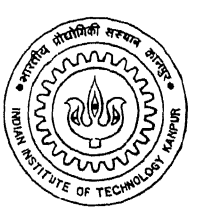
Planning and Synthesis of Hyper-Redundant Manipulators

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

by

Praveen Tayal



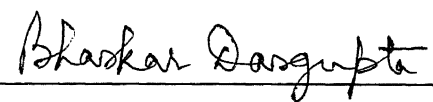
to the

Department of Mechanical Engineering
Indian Institute of Technology, Kanpur

February, 2000

Certificate

This is to certify that the work contained in the thesis entitled "*Planning and Synthesis of Hyper-Redundant Manipulators*", by *Praveen Tayal*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.



Dr. Bhaskar Dasgupta
Department of Mechanical Engg.,
Indian Institute of Technology,
Kanpur.

Acknowledgement

I express my sincere thanks to my thesis supervisor Dr. Bhaskar Dasgupta for his excellent guidance, invaluable suggestions, constant support and encouragement throughout this work.

I am grateful to Dr. A. Mukerjee for his teaching a course on Robot Motion Planning, which indeed helped me, to work with good fundamental background and concepts throughout this work. His timely appointments, kind treatment and valuable suggestions which always helped me a lot in making this work more effective.

I am indebted to Mr. D. K. Tamrakar, Tushar Goel and Mr. Sambhoo for their timely help in carrying out this work. I also appreciate the responsiveness of Amit, Anurag, Priyanshu, Sarvesh during my difficult days. I am very thankful to Ms. Pooja and Ms. Arlene for their moral support, love and affection. I thank Manish, Satya Prakesh, Shobhit, Jagdis, Kakkar, Arvind, PAV Prasad, Swaroop, Sanjay, Harish and Sreangshu for sharing thoughts and emotions with me. I thank Monica for designing my homepage on WWW and keeping my spirits high even in my most busy timings and making my stay at IIT-Kanpur an ever-memorable.

I specially acknowledge Ms. Raisinghani for sharing her precious time and support for almost all of my insurmountable problems. I thank very much my parents, sisters, brother, brother-in-laws for their love towards me. I wish to acknowledge all the friends and faculty members, who were a part of the process of intellectual enrichment and personal growth in many aspects of life.

And above all, I am thankful to the Almighty for enabling me to achieve higher goals.

Praveen Tayal

Abstract

In the recent developement of industrial robot manipulators, it is expected to add the sophisticated function of high performance motion. Hyper-redundant manipulators are one of powerful tools to realize that motion.

Robot manipulators which have more than the minimum number of degrees-of-freedom are termed as "kinematically redundant," or simply "redundant." Redundancy in manipulator design has been recognized as a means to improve manipulator performance in complex and unstructured environments. "Hyper-redundant" robots have a very large degree of kinematic redundancy, and are analogous in morphology and operation to snakes, elephant trunks, and tentacles. There are a number of very important applications where such robots would be advantageous.

Due to very large number of active degrees of freedom, the benifits of hyper-redundant robot include the ability to avoid obstacles, increased robustness with respect to mechanical failure, and the ability to perform new forms of robot locomotion and grasping.

In the work presented here, the author has first realized the motion planning of a hyper-redundant robot amid obstacles in a defined work-cell and then synthesized the manipulator, based on its kinematic parameters, to aid dexterity in reaching out almost all possible goal locations. The treatment has been restricted to the 2-D workspace.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Hyper-Redundant Manipulator

The term robot can convey many different meanings in the mind of the reader, depending on the context. In the treatment presented here, a robot will be taken to mean an industrial robot, also called a *robotic manipulator* or a *robotic arm*, in simple words it is an articulated robotic arm and is roughly similar to a human arm. It can be modeled as a chain of rigid links interconnected by flexible joints. The links corrospond to such features of the human anatomy as the chest, upperarm and forearm, while the joints correspond to the shoulder, elbow, and wrist. At the end of the robotic arm is an end-effector, also called a *tool gripper* or *hand*.

Robot manipulators have usually been designed to have no more than 6 degrees of fredom (DOF), which is the minimum DOF needed to perform the three-dimensional tasks. Although 6 DOF is sufficient from the conventional mechanical design viewpoint, it has limited the potential applications of robot manipulators. When a robot manipulator follows a given trajectory of the end-effector and needs to avoid hitting obstacle at the elbow, it requires more DOFs than it does when working in free workspace. A robot manipulator having more than 6 DOF is termed as a *redundant manipulator*, and if that degree of redundancy is too high (order of about 12-15 or more), it is called a *hyper-redundant manipulator*.

Conventional robots can reach a particular location with finite number of configurations. However, in a highly convoluted environment, a conventional robot will not be able to reach all positions without hitting an object [9]. A *hyper-redundant robot* can obtain these end-effector locations by flexing around multiple beams and columns. In order to do so the hyper-redundant robot by virtue of its large number of degrees of freedom must progressively move around the obstacles to get to its final location. In other words the hyper-redundant manipulator is no longer just reaching the desired location, but it is following a path to get there. This path is a successive set of configurations which bring the robot from an initial configuration to a final configuration with desired end-effector position and orientation.

Hyper-redundant robots offer advantages over traditional mobile robots and robot arms because they have enhanced flexibility and reachability, especialy in convoluted environments. These robots are well suited to inspect large space-fairing truss structures such as the future space station and can also be used to inspect any kind of leakage or breakage in a nuclear environment where a human being cannot sustain for long due to hazardous nuclear exposure. Serpentine mechanisms offer unique capabilities on earth to applications such as bridge inspection, search and rescue, surface coating, and minimally invasive surgery.

1.2 Scope of the Thesis

The work described in this thesis, exploits a geometric structure, termed *visibility graph*, a *roadmap*, to guide the motions of a *hyper-redundant robot* in highly convoluted spaces, with known obstacles and known target poses (locations to be serviced), then by using the information obtained by planning a detailed kinematic synthesis followed by mechanical design of the manipulator is performed. We also restrict our analysis for a single goal location in the work-cell. In practice we assume that there exists a finite number of goal locations in the work-cell.

Hyper-redundant robots are analogous in morphology to tentacles or **snakes.** Due to their high degree of articulation, hyper-redundant robots are potentially superior for operations in highly constrained and unusual environments encountered in applications such as inspection of nuclear installations and medical endoscopy. Although the large number of degrees of freedom of hyper-redundant

robots provide great functionality, they also pose a challenging research problem namely how to co-ordinate all of the actuators of the robot to yield motion.

1.3 Literature Review

Path planning for a robot (conventional manipulator as well as a polygonal robot) among fixed polygonal obstacles and various extensions of this basic problem have been studied during the past two decades. However, research strictly addressing manipulation planning is fairly recent. The first paper to tackle this problem is by Wilfong. It considers a single body robot translating in a 2D workspace with multiple movable objects. The robot and movable objects are modeled as convex polygons. The robot grasps an object by making one of its edges concides with an edge of the object. This definition of grasping extends to several movable objects. In fact, not much work has been reported related to planning and sythesis of hyper-redundant robot. There has been some path planning work on serpentine robots. Serpentine robots are like the hyper-redundant manipulator except its mobile base. So a serpentine robot can crawl like a snake. However, the planning schemes remain to be sensor based for even serpentine robot. The sensor based planning, however, assumes that there are perfect sensors on the robot, but this approach has not been implemented on the real robots. There has been little work devoted explicitly to serpentine motion planning. One approach is based on the definition of *tunnels* through an obstacle field, into which the manipulator "slithers" [10]. A local sensor based planning method was reported in Takanashi *et al.*, in which the snake robot maintains its end-effector location while it locally adatps to a time varying environment. In this method, the entire manipulator is fit within the tunnel and the part of the tunnel is continuously adapted away from any object that becomes unacceptably close. There has been a lot of work on the approaches which adapt the structure of a rigorous motion planning scheme to a sensor based implementation; one such scheme is based on a geometric structure, termed a *roadmap* [1]. Roadmaps are defined by following properties: *accessibility*, *connectivity*, and *departability*. These properties imply that the planner can construct a path between any two points in a connected component of the robot's free space by first finding a path onto the roadmap (accessibility), traversing the roadmap to the vicinity of the goal (connectivity), and then constructing a path from the roadmap to the global (departability).

Many mechanically redundant robots have been developed so far. Some were developed for research purposes. Some have already been used in real applications.

MELARM, with two, 7-DOF arms was developed at the Mechanical Engineering Laboratory, the Ministry of International Trade and Industry (MITI), Japan. Each arm has an anthropomorphic[1] structure. There was another 7-DOF manipulator named UJIBOT that has a nonanthropomorphic structure. UJIBOT was developed in 1979.at Kyoto University. This robot was used for experiments in obstacle avoidance. Another 7-DOF direct-drive manipulator was developed at MITI, Japan. This robot is intended for automation of sewing tasks in garment industries. Now coming to the redundant robots designed for practical use, there are several commercial welding robot system consisting of a 5- or 6-DOF arm and a 2-DOF position table. However, their redundancy has been used in a rather limited way. A 17-DOF robot system was designed for research of space, telerobotics developed by Robotics Research Co. The robot system consists of two 7-DOF arms and 3-DOF torso. .

For research on multi-arm co-ordination, two 6-DOF arms are often used. Several multi-fingured hand system have been developed for research on multi-finger co-ordination.

Whitney and Uchiyama, pointed out that a kinematically redundant robot manipulator having more than 6 DOF can be a remedy to overcome singularity[2]. The kinematic redundancy is also effective for enabling a robot manipulator to appraoch a workpiece from all direction avoiding obstacles in a workspace. Whiteny, discussed the redundancy of prosthetic arm. Although he suggested a criterion to minimize the integral of kinetic energy, he simplified the criterion due to the expected high computational cost, and instead proposed that we minimize the quadratic form of joint velocity instantenously. Nakano and Ozaki, on the other hand, proposed the minimum potential energy criterion, which imposes a constraint on the inverse kinematics of anthropomorphic manipulator so that elbow position stays as low as possible.

Liegeois discussed the active utilization of redundancy. He also showed the numerical simulation of utilizing redundancy for keeping the joint angles within their mechanical limitations. Hanafusa, Yoshikawa, and nakamura proposed a

[1]A serial-chain manipulator resembling a human arm

[2]The configurations, where the DOF degenerate and effectively become equivalent to less than the designed DOF, are called *singular points*

numerical algorithm to plan a joint trajectory of a redundant manipulator with limitations of joint angles and the torques and the geometric constraints imposed by obstacles.

In recent publications. Obstacle avoidance for kinematically redundant robots has been developed by C. Y. Chung, B. H. Lee and J. H. Lee , Automation and Systems Research Institute, Seoul National University, Korea. Sensor-based motion planning in three dimensions for a highly redundant snake robot by D. Reznik and V. Lumelsky. Experimental Results for Sensor Based Planning for Hyper-redundant Manipulators N. Takanashi, H. Choset, and J. Burdick IEEE/IROS Yokohama, Japan.

Existing work in automated synthesis for robots has focused on configuration synthesis rather than detailed electromechanical design. Much of this research addresses the problem of configuration a robot from a set of self-contained modules. Paredis used a distributed, agent-based genetic algorithm (GA) to create fault-tolerant serial chain manipulators from a small inventory of link and revolute joint modules.

Farritor et al, proposed a methodology for modular mobile robots. Chen and Burdick, propose a method for determining an optimal configuration of modular links and joints. An Assembly Incidence Matrix (AIM) is used to represent serial chain manipulators. It is a matrix representation of the mechanism graph. A GA is used to determine the optimal assembly based on how many of a small number of task points are reachable by each assembly. Chedmail and Ramstien, used a GA to determine the base position and type (one of several manipulators) of a robot to optimize workspace reachability. McCrea discussed the application of GA to selection of several parameters of a manipulator used in bridge restoration.

Kim and Khosla used a GA to synthesize the kinematic parameters of a spatial manipulator with revolue joints and links modeled by line segments. A multi-stage optimization process first chooses the number and orientation of joints, then link lengths and joint angles for small number of points along a trajectory. Constraints are gradually enforced to ensure continuity between task points, and then a kinematic controller generates joint angles for immediate task points.

Roston used a GA to create a 2D generalized frame walker, which is evaluated on simulated terrain. Motion plans are evolved concurrently with each mechanism, and a second GA is used to set the parameters for the first GA.

1.4 Organisation of the Thesis

A brief summary of each chapter follows :

Chapter 2 provides an overview of the basic understanding of configuration space of the robot manipulator in generalized workspace. This chapter is also a short summary of hyper-redundant manipulator kinematics. The Jacobian matrix and its computation, the definition of manipulability and redundancy. This chapter forms a firm ground for the rest of the work. The "configuration space" explained in this chapter is used extinsively in the following chapters.

Chapter 3 discusses path planning of manipulator using potential field approach. The problem is formulated in the framework of steepest decent principle.

Chapter 4 discusses the path planning approach based on the concept of visibility graph search method. Also the concept of task priority is introduced. This chapter unfolds the concept as how the hyper-redundant manipulator can be planned to move successfully in the cluttered workspaces.

Chapter 5 introduces the optimization of manipulator to make it dexterous in convoluted environment. It explores the Genetic Algorithm applied for the optimization. In this Chapter we finally take up the optimum kinematic parameters of a manipulator for a given work-cell.

Chapter 2

Hyper-Redundant Manipulator Planning

2.1 Configuration Space

Consider a simple manipulator $\mathcal{A}$ made of p rigid objects, i.e. *links*, $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \ldots, \mathcal{A}_p$. Any two links $\mathcal{A}_i$ and $\mathcal{A}_j$ may possibily be connected by a joint. For simplification, we assume that the joint is either a **revolute joint** or a **prismatic joint**. A revolute joint is a hinge that constraints the relative motion of $\mathcal{A}_i$ and $\mathcal{A}_j$ to be a rotation around an axis fixed with respect to both objects. A prismatic joint is a sliding connection that constrains the relative motion to be a translation along an axis fixed with respect to both objects. These are the most usual joints, and almost all other joints can be approximated and treated in much the same way as combination of revolute and prismatic joint. Here, in this work, for simplification the study has been restricted on a hyper-redundant manipulator having revolute joints only.

Let $\mathcal{F}_{\mathcal{A}_i}$ be the frame attatched to $\mathcal{A}_i, i \in [1, p]$, as defined in Appendix[A]. The configuration of $\mathcal{A}$ is a specification of the position and orientation of every frame $\mathcal{F}_{\mathcal{A}_i}, i = 1$ to p, with respect to $\mathcal{F}_{\mathcal{W}}$. However, the various joints impose constraints on a feasible configurations in configuration space C [see appendix A]. These constraints select out a subspace of lower dimension in C, which is the actual configuration space of $\mathcal{A}$.

Example : Consider two cases, which contain a planar two-joint manipulator arm. Fig. 2.1 connects the objects $\mathcal{A}_1$ and $\mathcal{A}_2$ of a manipulator arm $\mathcal{A}$, by a revolute joint and Fig. 2.2 connects them by first revolute and then a prismatic joint.

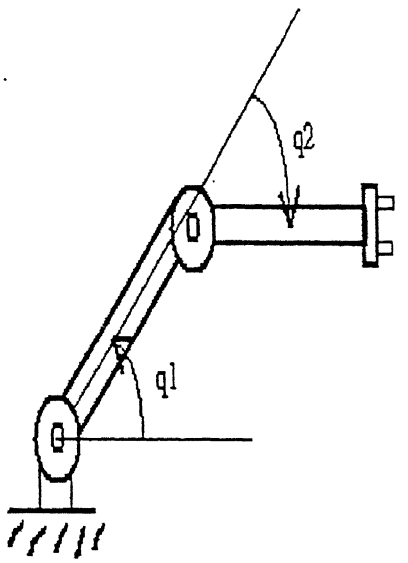


Figure 2.1: A planar two-revolute-joint manipulator

Consider Fig. 2.1, a planar two-revolute joint manipulator arm. The configuration space of this robot is a torus in a 3-D Euclidean space and a shell in 2-D space, iff the two angles q_1 and q_2 varies in $[0, 2\pi)$, with modulo 2π arithmetic.

While in the case of Fig. 2.2, a planar one-revolute-one-prismatic joint manipulator arm, the configuration space C-Space would be $C' = (R^2 \times SO(2))^2$ [see appendix A]. The first joint imposes two constraints expressing that the point about which $\mathcal{A}_1$ can rotate is fixed in the workspace, the second joint imposes two additional constraints on the position and orientation of $\mathcal{A}_2$ relative to $\mathcal{A}_1$. The four constraints are independent and determine a subspace C of dimension 2 in C'.

In this work, we have assumed that $\mathcal{A}$ is an articulated robot containing no closed kinematic loop and the workspace is two dimensional. Moreover,

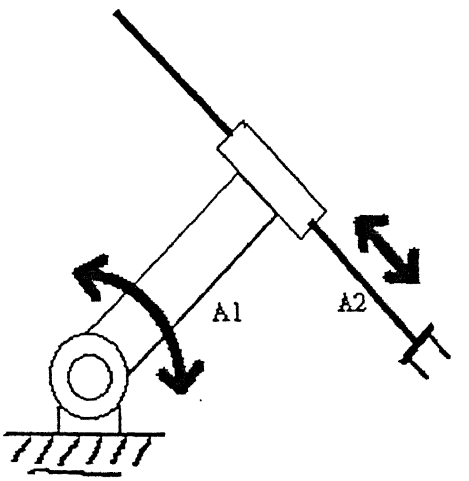


Figure 2.2: A planar one-revolute-one-prismatic joint manipulator

we have also assumed no branching of links as well as no mechanical stops in joints. Moreover self-body intersection amongst links is not penalized. If a hyper-redundanr robot consists of 15-20 links, its configuration space would be of order 15-20.

The concept of *configuration space* is used throughout this work in order to organise the various aspects of motion planning in a coherent frame-work. So configuration space is used here as a tool for formulating motion planning problem systematically.

2.2 Redundancy

If the number of joints (n) is greater than the dimension of manipulation variables (m), i.e. $n > m$, a manipulator is said to be a redundant manipulator, and if $n \gg m$ then it is called a **hyper-redundant manipulator** [5].

This manipulator is characterized by the fact that there exist infinite solutions of inverse kinematics.

Fig. 2.3 shows a 16-link manipulator, the angular displacements of each link has been taken with respect to the axis of orientation of its previous link. If we

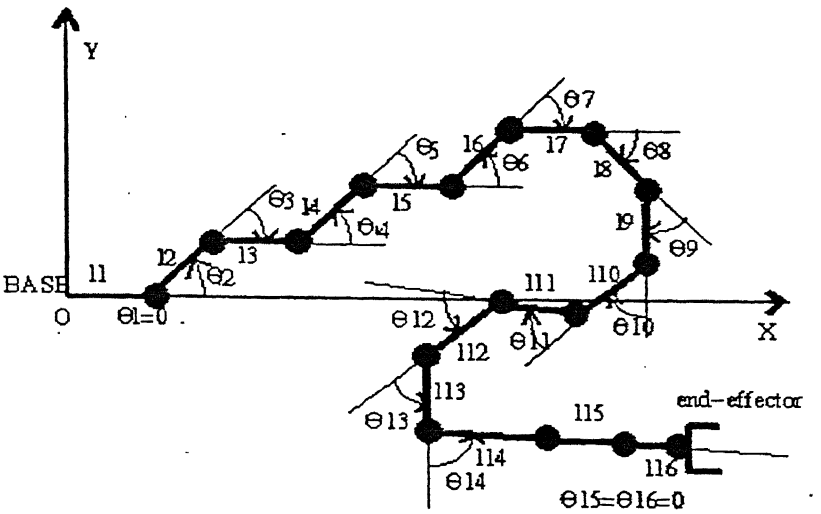


Figure 2.3: Hyper-redundant manipulator

explore this hyper-redundant manipulator into a 2-D work-cell then:

$$\begin{aligned} n &= 16 \\ m &= 2 \end{aligned} \tag{2.1}$$

Writing down the direct kinematics [2] for an end-effector point "A":

$$\begin{aligned} x &= l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) + \cdots + l_{16} \cos(\theta 1 + \theta_2 + \cdots + \theta_{16}) \\ y &= l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) + \cdots + l_{16} \sin(\theta 1 + \theta_2 + \cdots + \theta_{16}) \end{aligned}$$

Simplifying the equations, the position of end-effector in Fig. 2.3 are typically represented by two independent parameters, in 2-D workspace

$$\mathbf{r} = \begin{Bmatrix} x \\ y \end{Bmatrix} \in R^2$$

The relationship between $\mathbf{r}$ and the joint angle $\boldsymbol{\theta_i}$ of a robot manipulator is generally expressed by the following non-linear equation:

$$\boldsymbol{r} = f(\boldsymbol{\theta}) \in R^2, \quad \boldsymbol{\theta} \in R^n \quad \text{, where } n = 16 \text{ in this case.}$$

Differentiating we get

$$\delta r = J(\theta)\,\delta\theta \tag{2.2}$$

where $J(\theta) \in R^{m\times n}$ i.e. $R^{2\times 16}$ is called **Jacobian matrix** where δr is a linear

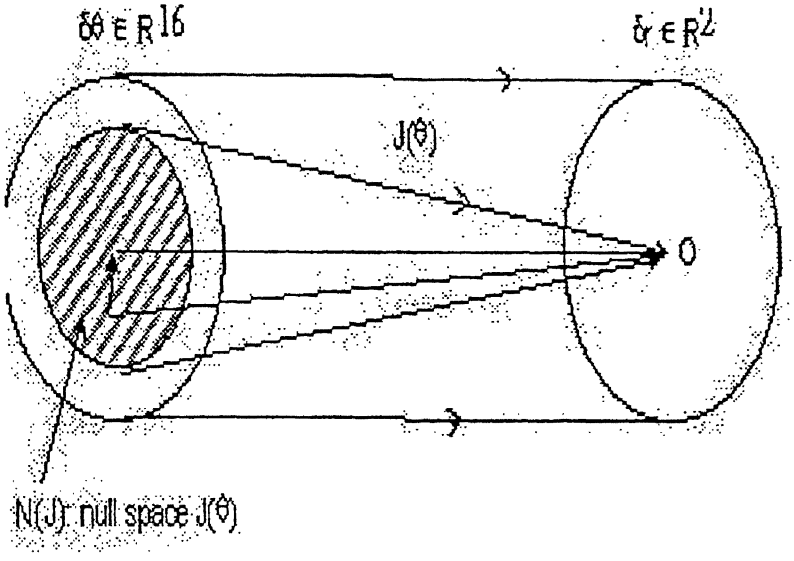


Figure 2.4: Differential motion mapping

mapping of $\delta\theta$ by Jacobian matrix. Fig. 2.4 illustrates the linear mapping by Eq. 2.1. So Eq. 2.1 can be regarded as a linear mapping from n-dimensional vector space R^n to m-dimensional space R^m. The subspace N(J) of Fig 2.4 is a null space of the linear mapping. Any element in this subspace is mapped into the zero vector in R^m : $J(\theta)\,\delta\theta = 0$. Therefore, any joint velocity vector $\delta\theta$ that belongs to the null space does not produce any velocity at the end-effector.

2.3 Path Planning Methods

There have been several path planning methods for an articulated robot, they include *Silhouette method, Collins decomposition, freeway methods, Approximate cell decomposition methods, Potential field approach* and *Sensor based planning.* (See Latombe [1])

All the methods mentioned above except potential field and sensor based planning, are limited to robots with relatively few joints (see Latombe [1]). So it is indeed very difficult to plan a path for a hyper-redundant robot, which due to large number of degrees of freedom, adds to the complexity in path planning.

However, potential field approach can be applied to articulated objects with minor adaptations. And sensor based planning has also shown good results, but this planning method is more realistic to be applied when the manipulator is in a dynamic work-cell. Here in this work two strategies for path planning have been implemented. These two approaches are :

1. Path planning using **potential field** approach.

2. Path planning using *visibility graph* method and then moving the manipulator along that obtained path using **priority based** approach .

Chapter 3

Path Planning using Potential Field Method

3.1 Potiential Field Approach

Typically, one can consider a collection of **control points** $a_j \in \mathcal{A}, j = 1, \ldots, Q$, which may be distributed over the various objects $\mathcal{A}_i$. Each point a_j is subjected to potential field $V_j(x)$ defined in the workspace. This potential will consists of an **attractive potential** depending upon the **goal** position of a_j in the workspace, and a **repulsive potential** depending upon the **obstacles** in the workspace.

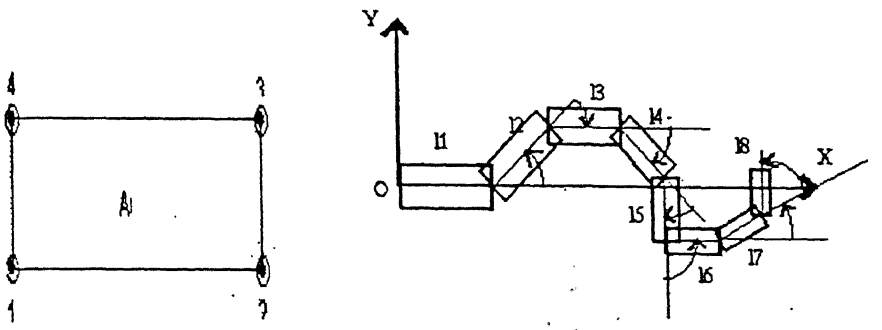
The potential function $U(q)$ in the robot configuration space is obtained by adding these potentials:

$$U(q) = \sum_{j=1}^{Q} V_j(a_j(q))$$

It gives rise to an attractive force defined by

$$\vec{F}(q) = -\vec{\nabla} U(q)$$

Here, we have assumed a link of a manipulator as a rectangular structure, as shown in Fig. 3.1(a). Corners of this rectangular link are encircled, to denote that they are the control points, which when distributed over the $\mathcal{A}_i$, give Fig. 3.1(b).



(a) A link having 4 control points

(b) The structure of the manipulator under observation

Figure 3.1: The geometric structure of a manipulator

So if there are eight links, for example, then the control points would be 32. Assuming links to be short along length, we can rely on the 4 control points in a link.

Let us consider, a robot configuration space C such that a configuration $\boldsymbol{q}$ is parameterizied by $(q_1, \ldots, q_p)$, with each q_i defining the relative position of two links connected by a joint, as described in section 2.1. The components of $\vec{\boldsymbol{F}}(\boldsymbol{q})$ induced are the **forces** applied at every revolute joint of the robot. In this work, the workspace considered being 2-D, we have $\mathcal{W} = R^2$, with x, y, being co-ordinates of a point $\mathbf{x} \in \mathcal{W}$ in $\mathcal{F}_{\mathcal{W}}$. Let $X_j(q_1, \ldots, q_p)$, $Y_j(q_1, \ldots, q_p)$ be the co-ordinates of $a_j(q)$ in $\mathcal{F}_{\mathcal{W}}$. The $i\text{-}th$ component of $\vec{\boldsymbol{F}}(\boldsymbol{q})$ is:

$$-\frac{\partial}{\partial q_i} U(q_1, \ldots, q_p) = -\sum_{j=1}^{Q} \left(\frac{\partial \mathrm{V_j}}{\partial x}\frac{\partial X_j}{\partial q_i} + \frac{\partial \mathrm{V_j}}{\partial y}\frac{\partial Y_j}{\partial q_i}\right)$$

Hence, we have :

$$\vec{\boldsymbol{F}}(\boldsymbol{q}) = -\sum_{j=1}^{Q} \boldsymbol{J}_j^T(\boldsymbol{q})\, \vec{\nabla}\, V_j(\boldsymbol{x})|_{\boldsymbol{x}=a_j(\boldsymbol{q})}$$

where $\boldsymbol{J}_j^T$ is a transpose of jacobian matrix $\boldsymbol{J}_j$ of the map :

$$(q_1, \ldots, q_p) \in R^p \rightarrow (X_j(q_1, \ldots, q_p), Y_j(q_1, \ldots, q_p)) \in R^2$$

3.2 Problem Formulation

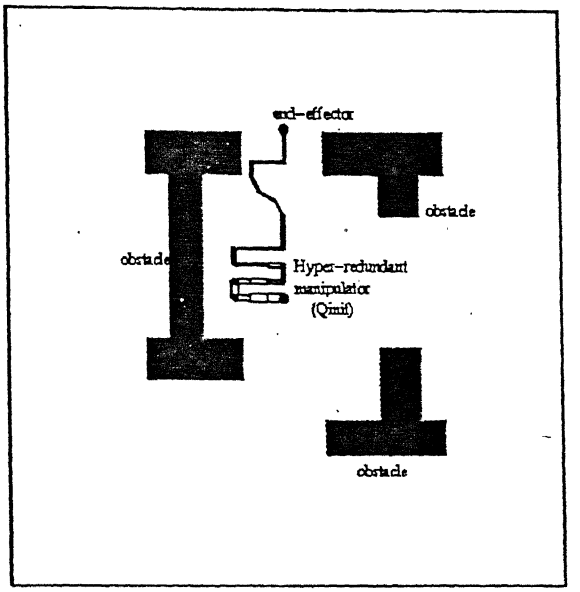


Figure 3.2: The work-cell

We consider a 2-D workspace, as shown in Fig 3.2, which is bounded by a polygonal object, a square in this case, called *room* and which contains the polygonal objects, treated as *obstacles*. A hyper-redundant robot is required to work in this type of work-cell.

For a hyper-redundant robot, which is considered to be a chain consisting of a large number of rectangular links of different sizes, the known parameters would constitute the work-cell, the manipulator initial configuration and the final goal point, to be reached.

Thus, the potential function of this hyper-redundant manipulator, can be assumed as the algebric sum of the attractive potential due to the goal and the repulsive potential depending upon the obstacles in the workspace, we call them $P_{attraction}$ and $P_{repulsion}$ respectively. So in this attempt, for the planning of HR manipulator, it has been assumed that $P_{attraction} \propto \sum_{j=1}^{Q} (Dis_{goal})_j^2$ and $P_{repulsion} \propto \sum_{j=1}^{Q} \sum_{i=1}^{n} \sum_{k=1}^{n_i'} \frac{1}{(Dis_{(obstacle_{i_k})}^2)_j}$. Where:

- n = the total number of obstacles and n_i' = the total number of vertices of the *i-th* polygonal obstacle.

- $(Dis_{goal})_j$ = the distance from a j-th control point on the manipulator to the goal point.
- $(Dis_{obstacle_{i_k}})_j$ = the distance from a j-th control point on a manipulator to the k-th point on the i-th ploygonal obstacle. This distance constitutes the shortest distance, so it may be either the distance from the manipulator control point to any of the vertices of a polygonal obstacle or the perpendicular distance to one of its edges.

In general,

$$P_{attraction} = \sum_{j=1}^{Q} K_1 (dis)_j^2 \tag{3.1}$$

where K_1 is a constant, and

$$P_{repulsion} = \begin{cases} \sum_{j=1}^{Q} K_2 (\frac{1}{dis_j} - \frac{1}{d_0})^2 & \text{if } dis_j \leq d_0 \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

where K_2 is a constant

$$P_{total} = P_{attraction} + P_{repulsion} \tag{3.3}$$

$$U(q) = P_{total}$$

$$\vec{F}(q) = -\vec{\nabla} U(q) \tag{3.4}$$

Eq. 3.4 determines the direction in which the end-effector of manipulator should move. Here, to make the computations further simple, the self-body potiential (repulsive potential) of the manipulator has not been considered, i.e. no mechanical stops, as mentioned earlier. This also means that self-intersections are not penalized.

3.3 Algorithm

In this algorithm, the planner is given a function (potential field) with a single global minimum at the goal configuration. It then attempts to connect $q_{initial}$

to q_{goal} by *down* motions. Each Down motion "descends" along $U(q)$ until it reaches a local minimum. Infact the construction of a navigation function (potential function) with no *local* minimum other than the goal configurations, is a difficult problem. That problem has a known solution only when C-obstacles have simple shapes. In this work we present a planner, which is a simple path planner, means which does not take care of local minimum, it does not avoid it.

In this algorithm, the input consists of the manipulator initial configuration, goal location, obstacles placement and the work-cell boundaries which is two dimensional, we name the close work-cell boundaries as *room*. So the internal boundaries of room also work as obstacles since the manipulator is neither allowed to cross them nor to hit them.

Major modules of the algorithm are the following :

- MakeManipulator() : This function, with known values of the structural parameters of a manipulator i.e. link lengths and joint angles, sets the manipulator configuration with respect to the base frame.
- PTotal() : This function defines the total potential defined in Eq .2.3
- JointAngleChange() : This function defines the actual change in the joint angles of the manipulator so that it can avoid obstacles while approaching the goal. It involves the numerical differentiation of the potential function as

$$\frac{\partial}{\partial q_i} U(q_1, \ldots, q_p) = - \lim_{\Delta q_i \to 0} \frac{Pot_{new} - Pot_{old}}{\Delta q_i}$$

  Where Δq_i denotes a very small and temperory change in q.
- NewJointAngle() : This function gives the successive configurations of manipulator in the direction of path generation.

```
procedure PathPlanning (q_init, q_goal, q_obstacles)
begin
install q_init to q_mani
while Dis is greater than EPSILON do
      begin
            MakeManipulator();
            PTotal();
            JointAngleChange();
            NewJointAngle();
```

```
            Calculate the new distance from the manipulator end-effector point
            to the goal point which is equal to the Dis;
        end;
end;
```

3.4 Results

A few examples are shown for a work-cell having fixed obstacles and manipulator initial configuration but different goal locations.

Here the inputs are as follows.

Work-cell : A square of sides of length 35 units.

Obstacles : Three obstacles as shown in Fig. 3.3(a).

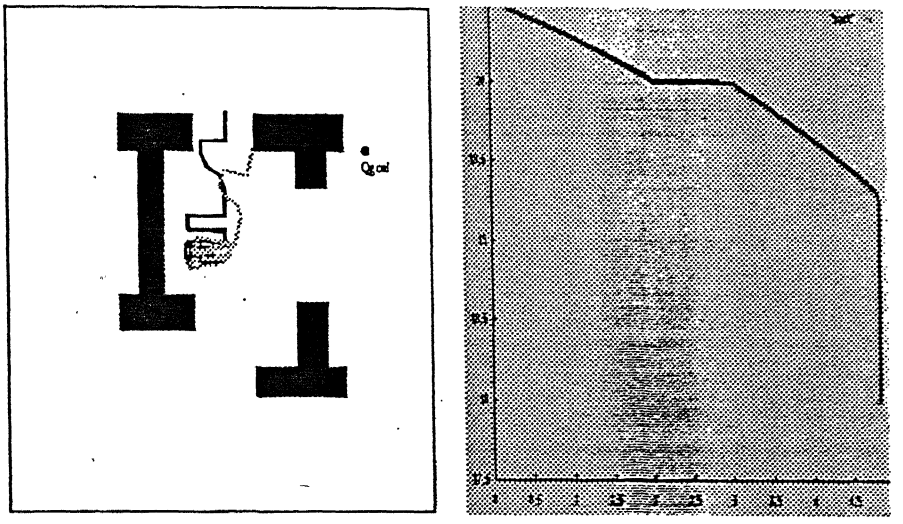
Manipulator : The robot manipulator having 25 links, each link has a react-angular dimension with link-length 2 units and the link-widths are taken to be in geometric progression (GP) starting as 1 unit for the base link and going according to the GP factor .95 for each succssesive link. The position of end-effector is at (0,20).

Initial Configuration : The initial configuration of the manipulator as shown in dark black in the figure with end-effector placed at co-ordinates (0,20). The coordinate frame is fixed at the base of the manipulator.

Different goal locations have been excercised taking the same inputs in the following cases. Case 1: (25,15), Case 2: (-22,13), Case 3: (-20,-25), Case 4: (25,0), Case 5: (-20,25), Case 6: (20,-25).

In all the cases shown in Fig. 3.3-3.8, the filled objects denote the obstacles, the darker presentation of manipulator shows the initial configuration of the manipulator and the light one shows the final configuration of the same manipulator. Fig. 3.3, Fig. 3.4, Fig. 3.5 and Fig. 3.8 show the inability of the manipulator to reach to the goal even if the path exists[1]. Instead, it gets trapped in the **local** minimum. The corresponding path of the end-effector of the manipulator is shown in the right hand side figure for each case. Fig. 3.6 and Fig. 3.7 show the success of the manipulator in reaching to its goal.

[1]In chapter 4, we will see the success of manipulator in reaching to the goal for all the unsuccessful cases here



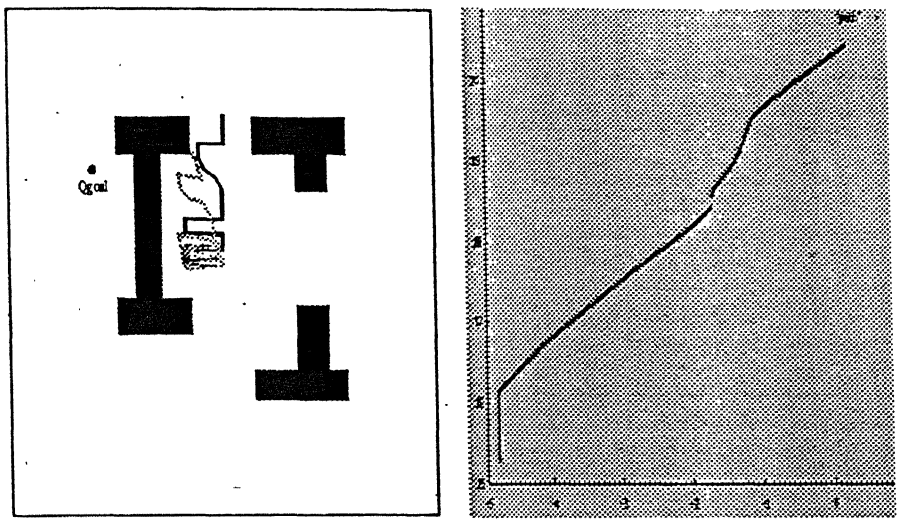
(a) Workcell

(b) Path starts from point (0,20) and ends up at point (4.7,18), manipulator fails to reach the goal

Figure 3.3: Path planning using "potential field method" : Case 1

3.5 Conclusion

In this chapter the path planning strategy using potential field method has been presented. This technique simply follows the steepest decent of potential function until the goal configuration is attained. However, it may get stuck at a minimum of the potential other than the goal configuration. Dealing with the local minimum within this planning is not simple. First, the fact that the local minimum has been attained must be recognized. Since motions are discretized, the planner usually does not stop excatly at zero-force configuration; instead, it typically generates segments "looping" around this configuration. This may be detected by checking that several successive configurations q_i's are not too close to each other. Second, the planner must escape the local minimum. This can be made possible by generating random motions and keep tracking at every random motions till we find those which helps the manipulator to escape from the local minimum. But that itself consumes lots of time and finally we may fail to obtain a path even if the path exists. It is very well evident from the results that potential field method for path planning, without Randomized motions, is

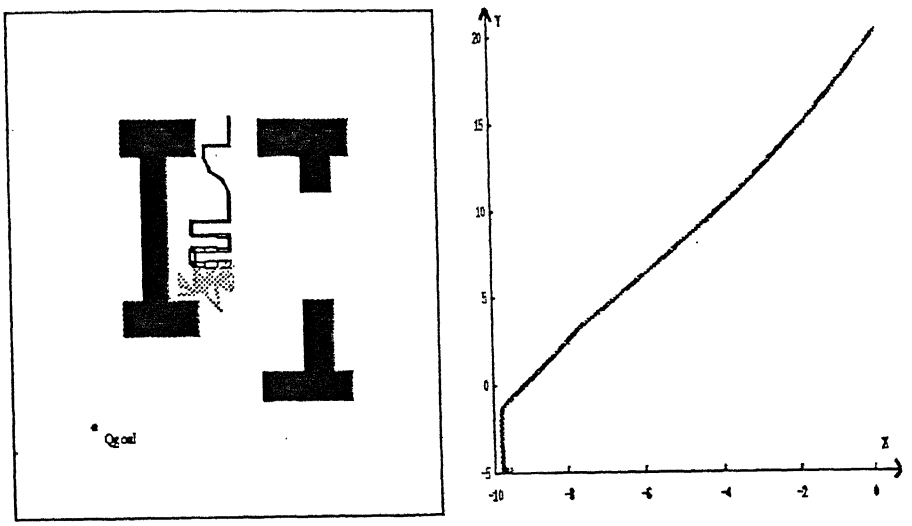


(a) Workcell

(b) Path starts from point (0,20) and ends up at point (-4.8,15.3), manipulator fails to reach the goal

Figure 3.4: Path planning using "potential field method" : Case 2

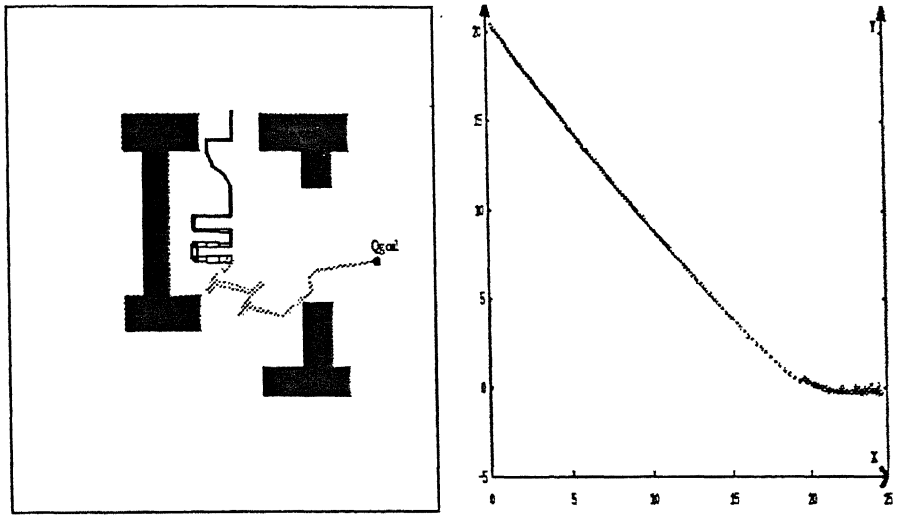
reliable when the workspace environment is not very cluttered. Moreover we have not taken up the self-intersection of the links into consideration. The self-body intersection may be avoided in much better way if we give the positive potential to the links of the manipulator, so whenever a link comes very close to any of the other links it gets pushed away from the other links due to repulsion force.



(a) Workcell

(b) Path starts from point (0,20) and ends up at point (-9.8,-5), manipulator fails to reach the goal

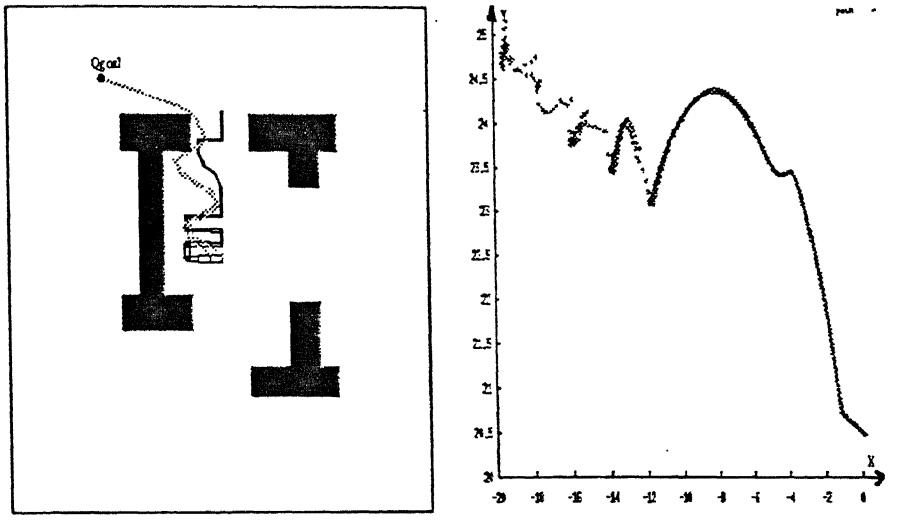
Figure 3.5: Path planning using "potential field method" : Case 3



(a) Workcell

(b) Path starts from point (0,20) and ends up at point (25,0), manipulator reaches the goal

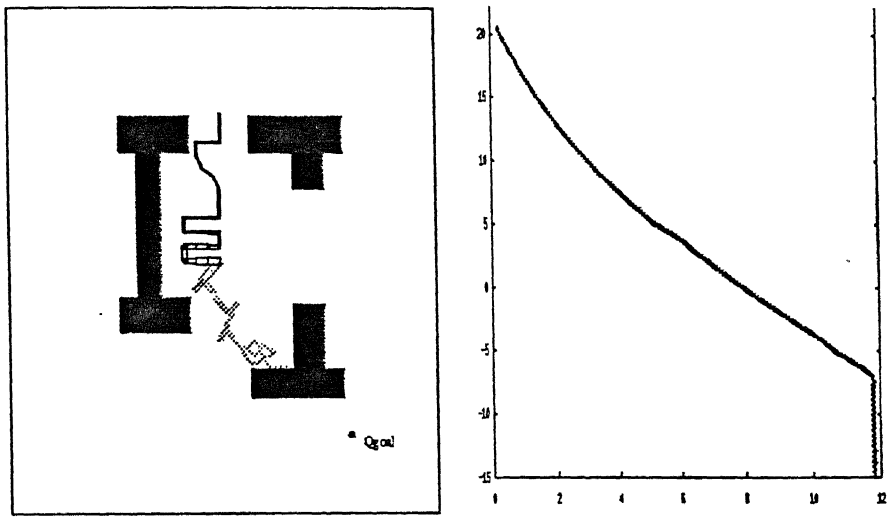
Figure 3.6: Path planning using "potential field method" : Case 4



(a) Workcell

(b) Path starts from point (0,20) and ends up at point (-20,25), manipulator reaches the goal

Figure 3.7: Path planning using "potential field method" : Case 5



(a) Workcell

(b) Path starts from point (0,20) and ends up at point (12,-15), manipulator fails to reach the goal

Figure 3.8: Path planning using "potential field method" : Case 6

Chapter 4

Path Planning with Order of Priority

In this method, a task is divided into subtasks according to the order of priority. Then the joint motion is resolved such that the subtasks with lower priority are realized using redundancy or extra DOF, not committed to satisfying the subtask requirements of higher priority.

In this chapter, the manipulator is planned so that it first plans the path from its initial configuration to the goal point, using a *roadmap* planning method. And then follows that path avoiding obstacles in the work space. So this planning work is two fold. First planning a path, if possible, and then following along that path avoiding obstacles.

4.1 Path Planning using Visibility Graph Search Method

The first goal is to find the feasible path from q_{init} to q_{goal}. For that purpose a *roadmap* method, called **visibility graph** method has been used.

The princilpe of visibility graph method is to construct a semi-free path [Appendix A] as a simple polygonal line connecting the initial configuration q_{init} to the goal configuration q_{goal} through vertices of CB and edges of CB.

To generate a path using this method, the manipulator is assumed as a point robot placed at the initial configuration of the end-effector point. see Fig. 4.1.

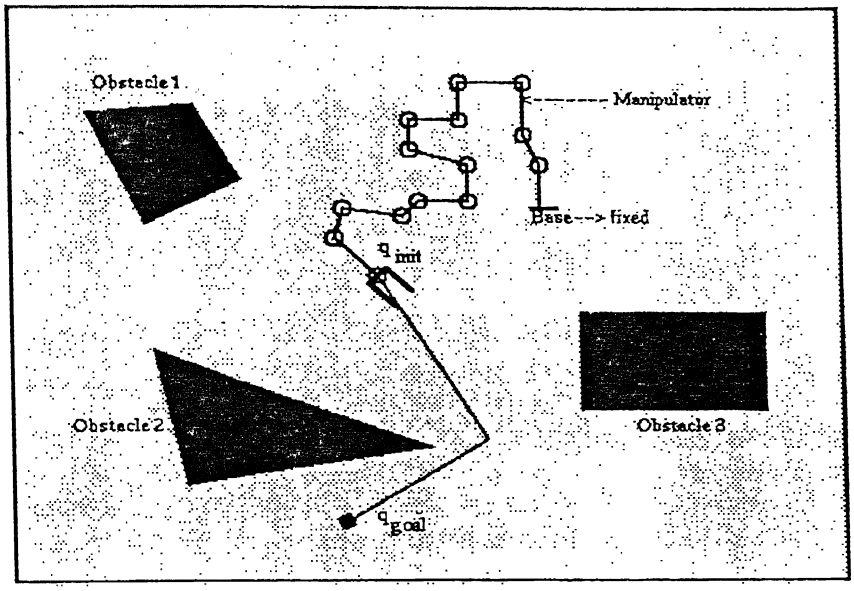


Figure 4.1: Work-cell with obstacles and a manipulator with path connecting q_{init} and q_{goal}

4.1.1 Problem Formulation

Let $\mathcal{A}$ be a single point object – the robot – moving in a 2-dimensional workspace $\mathcal{W}$, represented as R^2.

Let $B_1, B_2, \ldots, B_q$ be the fixed rigid objects distributed in $\mathcal{W}$. The B_i's are called obstacles, also include the room interior boundaries (wall).

Assuming that both the geometry of $B_1, \ldots, B_q$ and the location of B_i's in $\mathcal{W}$ are accurately known. Assume furthur that no kinematic constrained limit the motion of that assumed point robot $\mathcal{A}$. The problem is : Given an initial position and goal position of $\mathcal{A}$ in $\mathcal{W}$, generate a path τ specifying a continous sequence of positions of $\mathcal{A}$ avoiding contact with B_i's, starting at the initial position and terminating at the goal position. Report failure if no such path exists.

Being a point robot the $\mathcal{CB}$ would have the same geometry as B_i's have, in the given work space, so then the generated path using visibility graph method would yield the path which would actually touch the obstacles boundaries. So if we allow this path which goes touching the obstacles boundaries, we would not be fulfilling the our requirement to avoiding obstacles. To come over this problem, enlargement [Appendix B] (scaling) of polygonal obstacles has been

used and finally after applying this method on several cases it has been taken as an acceptable method. The enlargement of obstacles (B_i's) also includes the contraction of the work space boundaries i.e *room* boundaries.

After enlargement of the obstacles we use the enlarged obstacles data as an input for visibility graph method to generate a feasible path. Using this strategy of scaling of obstacles, it is quite possible that the goal locations which are very near to obstacles may get in-accessible and finally the visibility graph search ends up with no path, even if there exists a path.

4.1.2 Visibility Graph Search Algorithm

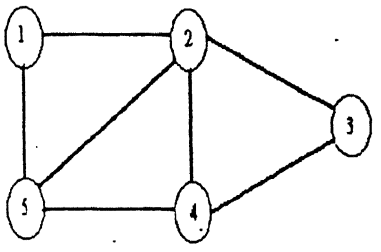
There are two standard ways to represent a graph G = (V,E): where V represents the vertex ponits and E represents the edge connecting two points (V's) , they are, first as a collection of adjacency list and as an adjacency matrix. In this work, we have used the adjacency matrix representation to find connectivity between points. In that representation, we assume that the vertices are numbered $1, 2, ..., V$ in some arbitrary manner. the adjacency matrix representation of a graph G then consists of a $V \times V$ matrix $A = a_{ij}$ such that :

$$a_{ij} = \begin{cases} dist_{ij} & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

Fig. 4.2(b) shows the adjancency matrix [4] of the undirected graph in Fig. 4.2(a). So the first objective is to find out the connectivity graph between the given obstacles vertex points and goal position and the assumed point robot.

In Fig. 4.3(a). we have 4 obstacles having total number of obstacles vertices equal to 5+4+4+3 = 16 with one initial position and one goal position. So total number of vertices which are to be connected to build a connectivity graph are equal to 18, including initial position and goal position. Then using the visibility graph method, we get the shortest path, which is shown by the dark black line amongs all other path, shown by grey lines. The obstacles are filled with black colour. Similarly for Fig. 4.3(b).

To compute the adjacency matrix in simplified manner, we find the connectivity graph between **tangent points**. Let X be a vertex of CB and L be a straight line passing through X. L is **tangent** to CB at X if and only if in a neighborhood $\mathcal{U}$ of X the interior of CB lies entirely on a single side of L. The visibility graph



	1	2	3	4	5
1	0	dis_12	0	0	dis_15
2	dis_21	0	dis_23	dis_24	dis_25
3	0	dis_32	0	dis_34	0
4	0	dis_42	dis_43	0	dis_45
5	dis_51	dis_52	0	dis_54	0

dis_ij = distance from ith node to jth node = dis_ji

(a) (b)

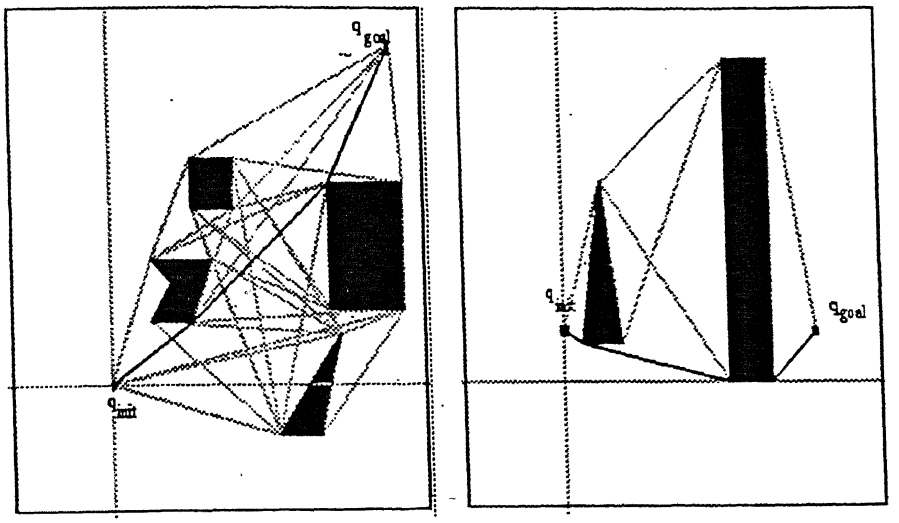
Figure 4.2: Two representations if an undirected graph [4]. (a) An undirected graph G having 5 vertices and seven edges. (b) The adjacency-matrix representation of G.

so obtained is called the **reduced visibility graph**. Fig. 4.4. shows the reduced visibility graph where the plain lines are the tangent segments and the dashed lines denote the non-tangent segments between the vertices of the polygonal obstacles.

We have used the A^* algorithm [1], which guarantees to return a path of minimum cost (distance) whenever a path exists, and to return failure otherwise. The input consists of the adjacency matrix (we call it cost[i][j]), N_{init} and N_{goal}. The cost[i][j] matrix gives us two informations, first, the connectivity between different nodes, i.e the un-directed graph G, and second, the cost (distance) function. All the nodes in G are first marked *unvisited*. The algorithm makes use of a list denoted by OPEN that contains nodes of G sorted by the values of the cost (distance) function.

The list OPEN supports the following operations:

- FIRST(OPEN) : remove the node of OPEN with the smallest value of cost[i][j] and return it.



(a) (b)

Figure 4.3: Finding shortest path using visibility graph method. (a) The shortest path is denoted by the dark black line joining from q_{init} to q_{goal}, similarly for another work-cell (b).

- INSERT(N, OPEN) : insert node N in OPEN.
- MEMBER(N, OPEN) : evaluate to true if N is in OPEN and false otherwise.
- DELETE(N, OPEN) : remove node N from open.
- EMPTY(OPEN) : evaluate to true if OPEN is empty and to false otherwise.

A^* explores G iteratively by following paths originating at N_{init} i.e. the initial position of the point robot. At the begining of every iteration, there are some nodes that the algorithm has already visited, and there may be others that are still unvisited. For each visited node N the previous iterations have produced one or several paths connecting N_{init} to N, but the algorithm only memorizes a representation of a path of minimum cost, i.e. the path of minimum distance among those so far constructed. At any instant, the set of all such paths forms

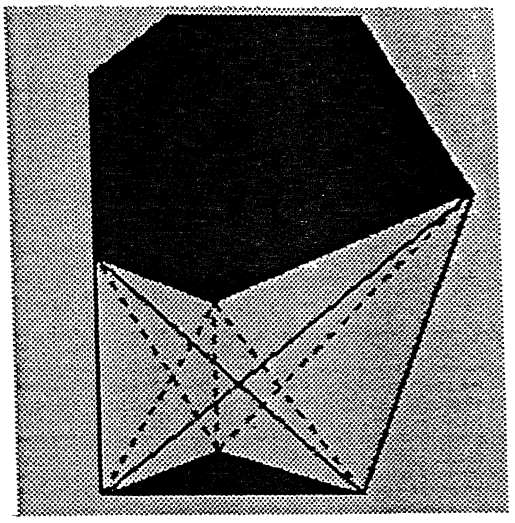


Figure 4.4: This figure illustrates the concept of a tangent segment between two polygons

a spanning tree T of the subset of G so far explored. T is represented by associating to each visited node N (except N_{init}) a pointer to its parent node in the current T.

A^* assigns the cost function [9] i.e. the distance function g(N) to every node N in current T. This function is a estimate of cost of the minimum cost path in G connecting N_{init} to N_{goal} and constrained to go through N.

At each iteration, A^* explores the nodes adjacent to the node returned by FIRST (OPEN). Initially both the tree T and the list OPEN are empty.

```
procedure A*(cost[][], N_init, N_goal)
        begin
            install N_init into T;
            INSERT(N_init, OPEN); mark N_init visited;
            while → EMPTY(OPEN) do
                begin
                    N ← FIRST(OPEN);
                    if N == N_goal then exit while loop;
                    for every node N' adjacent to N in G do
                        if N' is not visited then
                            begin
                                add N' to T with a pointer towards N;
                                INSERT(N', OPEN); mark N' visited;
```

```
                end;
            else if g(N') > g(N) + cost[N][N'] then
                begin
                    modify T by redirectiing the pointer of N' towards N;
                    if MEMBER(N', OPEN) then DELETE(N', OPEN);
                    INSERT(N', OPEN)
                end;
        end;
    if ↛ EMPTY(OPEN) then
        return the constructed path by tracing the pointers in T from N_goal
        back to N_init;
    else return Failure;
end;
```

Once the path is found, it is now needed for the manipulator to follow that path avoiding obstacles. Now we bring the task with priority based method [5] to move the hyper-redundant robot to follow the generated path.

The first priority subtask for the manipulator is to follow the path in small incremental steps, and the lower priority subtask is to avoid obstacles while following the path, which is realized using redundancy or extra DOFs. so before going into the details, let us look at its general formulation:

4.2 Inverse Kinematics considering the Order of Priority

General Formulation

First we assume that a task is composed of two subtasks (it may go to any number) with the order of priority. The first priority subtask is specified using the first manipulation variable, $\boldsymbol{r_1} \in R^{m_1}$ and the second priority subtask by second manipulation variable, $\boldsymbol{r_2} \in R^{m_2}$. The kinematic relationship between joint variables $\boldsymbol{\theta} \in R^n$ and the manipulation variables are expressed as

$$\boldsymbol{r_i} = \boldsymbol{f_i}(\boldsymbol{\theta}) \quad (i = 1, 2) \tag{4.1}$$

There differential relationships are derived as

$$\dot{\boldsymbol{r}}_i = \boldsymbol{J_i}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \quad (i = 1, 2) \tag{4.2}$$

Where $J_i(\theta) = \partial f_i/\partial\theta \in R^{m_i \times n}$ is the Jacobian matrix of the i-th manipulation variable.

With $m_1 < n$ Eq .4.2 for $i = 1$ generally has an infinite variety of solutions of θ whose general solution is obtained using as follows

$$\dot{\theta} = J_1^{\#}(\theta)\dot{r}_1 + (E_n - J_1^{\#}(\theta)J_1(\theta))y \tag{4.3}$$

Where $J_1^{\#}$ is called the pseudo-inverse of Jacobian matrix J_1(see Appendix C), $y \in R^n$ is an arbitrary vector. and $E_n \in R^{n\times n}$ indicates an identity matrix. If the exact solution does not exists, Eq. 4.3 covers all the least-suqares solutions that minimize $|| \dot{r}_1 - J_1(\theta)\dot{\theta} ||$.

Now, substituting Eq. 4.3 into Eq. 4.2 for i=2, we have the following equations :

$$J_2(E_n - J_1^{\#}J_1)y = \dot{r}_2 - J_1J_1^{\#}\dot{r}_1 \tag{4.4}$$

If the exact solution of y exists for Eq. 4.4 it implies that the second manipulation variable can be realized. The exact solution however does not generally exists. We can obtain y that maintains and minimizes $|| \dot{r}_2 - J_2(\theta)\dot{\theta} ||$, in the same fashion as Eq. 4.3, that is :

$$\begin{aligned} y &= \hat{J}_2^{\#}(\dot{r}_2 - J_2J_1^{\#}\dot{r}_1) + (E_n - \hat{J}_2^{\#}\hat{J}_2)z \\ \hat{J}_2 &= J_2(E_n - J_1^{\#}J_1) \end{aligned} \tag{4.5}$$

Where $z \in R^n$ is an arbitrary vector. The solution θ is obtained from Eq. 4.3 and Eq. 4.5 as follows :

$$\begin{aligned} \dot{\theta} = {} & J_1^{\#}\dot{r}_1 + (E_n - J_1^{\#}J_1)\hat{J}_2^{\#}(\dot{r}_2 - J_2J_1^{\#}\dot{r}_1) + \\ & (E_n - J_1^{\#}J_1)(E_n - \hat{J}_2^{\#}\hat{J}_2)z \end{aligned} \tag{4.6}$$

It has been proved that the second term on R.H.S. of Eq. 4.6 is reduced to $\hat{J}_2^{\#}(\dot{r}_2 - J_2J_1^{\#}\dot{r}_1)$. The proof considers the symmetry and idempotency[1] of $(E_n - J_1^{\#}J_1)$. Hence Eq. 4.6 becomes

$$\dot{\theta} = J_1^{\#}\dot{r}_1 + \hat{J}_2^{\#}(\dot{r}_2 - J_2J_1^{\#}\dot{r}_1)(E_n - J_1^{\#}J_1)(E_n - \hat{J}_2^{\#}\hat{J}_2)z \tag{4.7}$$

[1]Matrix M is called *idempotent* if it satisfies $M^2 = M$

Eq. 4.7 represents the inverse kinematic solution taking account of the priority of subtasks.

We define the range space of the Jacobian matrix, $R(J)$, as the manipulable space, and the null space of the Jacobian matrix, $N(J)$ (see section 2.2), as the redundant space. Fig. 4.5 shows the general relationship between the manipulable and redundant spaces for the first and second manipulation variables. In Fig. 4.5, subspace A, subspace B, and subspace C are $S_{R1}^{\perp}$, $S_{R1} \cap (S_{R1} \cap S_{R2})^{\perp}$ and $S_{R1} \cap S_{R2}$, respectively, where $S_{Ri} = N(J)$ and $S^{\perp}$ indicates the orthogonal component of subspace S.

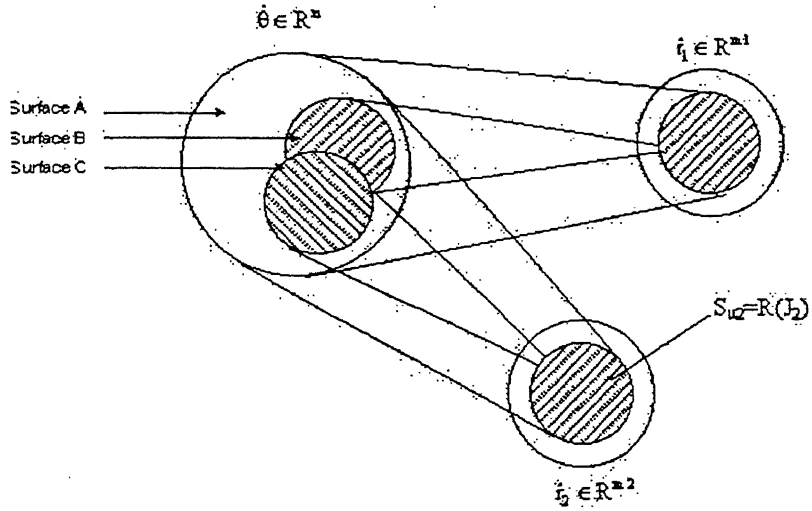


Figure 4.5: Priority mapping

Subspace A allows all possible contributions of $\dot{\theta}$ to the first manipulation variable. Subspace B implies the contributions of $\dot{\theta}$ to the second manipulation variable. Subspace C shows the remaining DOF that affect neither the first nor the second manipulation variables. Subspace C can be used for third or higher manipulation variables if necessary.

The first term in the right hand side of Eq. 4.7 is the least-squares mapping of $\dot{r}_1$ onto subspace A. The second term implies the least-square mapping of $\dot{r}_2 - J_2 J_1^{\#} \dot{r}_1$ onto subspace B, where $\dot{r}_2 - J_2 J_1^{\#} \dot{r}_1$ is the modified desired value of the second variable due to the effect of the first term on the second manipulation variable.

In the case of $r_2 = \theta$ Eq. 4.7 can be reduced to a simpler form. The term corrsponding to the third term of Eq. 4.7 becomes intrinsically equal to zero, which means that zero DOF remains for the higher manipulation variables, because the second manipulation variable $r_2 = \theta$ requires all the DOF remaining after being used for r_1 .

so from Eq.4.5, if we put $r_2 = \theta$, we get $J_2(\theta) = E_n$

$$\begin{aligned}
\hat{J}_2(\theta) &\triangleq J_2(E_n - J_1^{\#}J_1) \\
&\triangleq E_n(E_n - J_1^{\#}J_1) \\
&\triangleq E_n - J_1^{\#}J_1
\end{aligned}$$

$$\begin{aligned}
\hat{J}_2^{\#}(\theta) &\triangleq \hat{J}_2^{\top}(\hat{J}_2\hat{J}_2^{\top})^{-1} \\
&\triangleq (E_n - J_1^{\#}J_1)^{\top}((E_n - J_1^{\#}J_1)(E_n - J_1^{\#}J_1)^{\top})^{-1} \\
&\triangleq (E_n - J_1^{\#}J_1)(E_n)^{-1} \\
&\triangleq E_n - J_1^{\#}J_1
\end{aligned}$$

Substituting these reduced forms of J_2, $\hat{J}_2$ and $\hat{J}_2^{\#}$in Eq.4.7 with no third variable z i.e. $z = 0$, we get

$$\begin{aligned}
\dot{\theta} &= J_1^{\#}\dot{r}_1 + (E_n - J_1^{\#}J_1)(\dot{r}_2 - E_nJ_1^{\#}\dot{r}_1) + 0 \\
&= J_1^{\#}\dot{r}_1 + E_n\dot{r}_2 - J_1^{\#}\dot{r}_1 - J_1^{\#}J_1\dot{r}_2 + J_1^{\#}J_1J_1^{\#}\dot{r}_1 \\
&= J_1^{\#}\dot{r}_1 + (E_n - J_1^{\#}J_1)\dot{r}_2 \quad \text{where } J_1J_1^{\#}J_1 = E_n
\end{aligned} \tag{4.8}$$

We make use of Eq. 4.8, utilizing redunancy to avoid obstacles in the workcell. We use the arbitrary vector y from the potential function that takes larger values as the manipulator approaches towards the obstacles in the region. The potential approach is described by :

$$\dot{\theta} = J_1^{\#}\dot{r}_1 + (E_n - J_1^{\#}J_1)(-k\frac{\partial P}{\partial \theta})^{\top} \tag{4.9}$$

where k is a positive constant.

The potential function is defined by

$$P = \begin{cases} k_0 \sum_{i=1}^{n}(D_i)^{-1} & \text{if } D_i < d_0 \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

where k_0 is a constant, and d_0 is the distance above which the effect of potential function is neglected.

where P is the potential function due to obstacles, here, no constraints due to joint limits have been considered, this also means that the selfbody intersection is not penalized. The $X_i \triangleq (x_i, y_i)^\top$ (units) are positions of points on the manipulator and are used to evaluate the distance D_i between mainpulator and the obstacles. The locations of these points are indicated in Fig. 2.3.

Once the path is obtained using visibility graph search, it is then this priority task method which realizes the end-effector of the manipulator to move along that path while rest of the remaining DOF (links) are used to avoid obstacles.

4.3 Algorithm

Step1

First we calculate the shortest path using 2-D visibility graph search method as described in Section 4.1

Step2

Then we fragment that shortest path into various small steps which actually give $\delta r_1 = \begin{Bmatrix} \delta x \\ \delta y \end{Bmatrix}$ where $\delta x = x_{i+1} - x_i$ and $\delta y = y_{i+1} - y_i$ and the r_1 is the first priority manipulable variable. So the end-effector is supposed to move along that path forwarding itself on these steps, from the given initial position to the goal position.

Step3

Then by using Eq. 4.9, we finally use the extra DOF to avoid the obstacles around while letting the end-effector to follow the path, obtained by step 1.

4.4 Results

Case 1 : initial position = (0,20), goal position = (25,15), The shortest path obtained by visibility graph search method is : $(0, 20) \rightarrow (0, 26.3) \rightarrow (22.38, 26.3) \rightarrow$

(25, 15). The room dimensions are taken to be a square having each side 35 units. The robot manipulator base is kept at (0, 0). The number of links in the hyper-redundant manipulator are taken as 25 links each having length 2 units. The initial configuration of the manipulator is as shown in Fig. 4.6(a). In Fig. 4.6(a), the end-effector position is shown by a thick point which is placed at (0, 20). Here the coordinate frame is fixed at the centre of Room and the base of the manipulator is kept at origin. This case shows the success of path planning of the manipulator using priority based method, while the same case showed failure when we applied the potential field approach (see Case 1, Fig. 3.3).

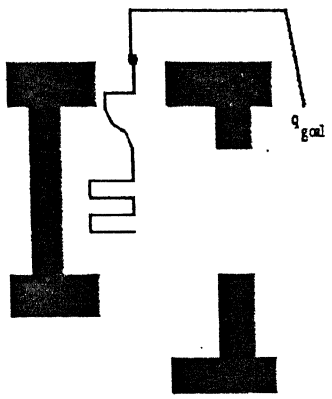


Fig.4.6(a)

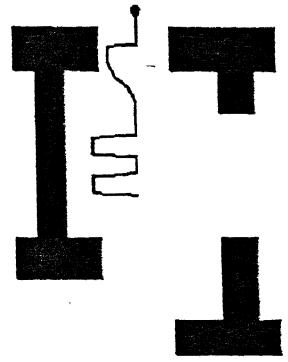


Fig.4.6(b)

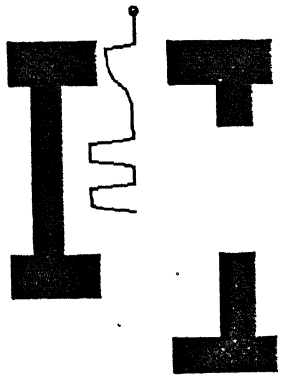


Fig.4.6(c)

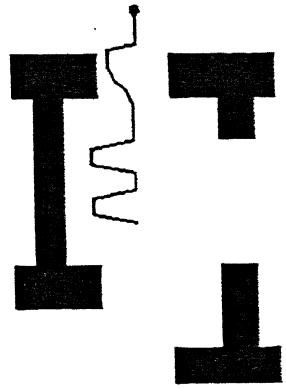


Fig.4.6(d)

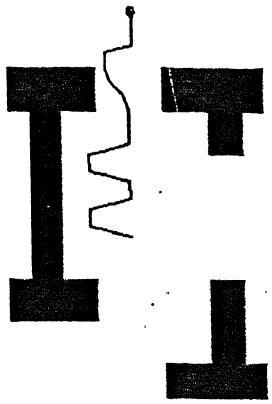


Fig.4.6(e)

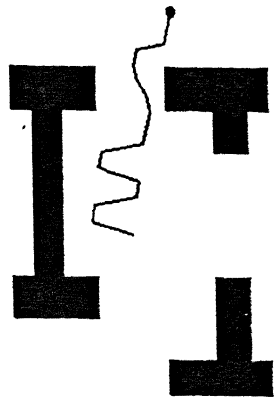


Fig.4.6(f)

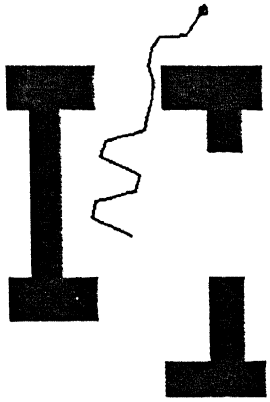


Fig.4.6(g)

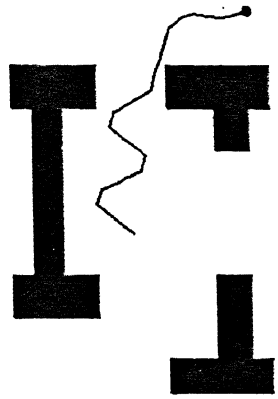


Fig.4.6(h)

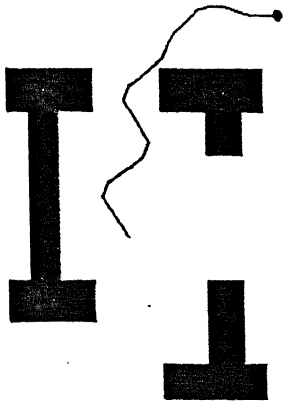


Fig.4.6(i)

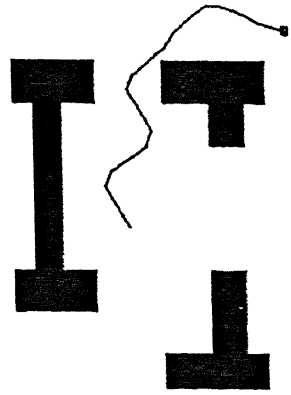


Fig.4.6(j)

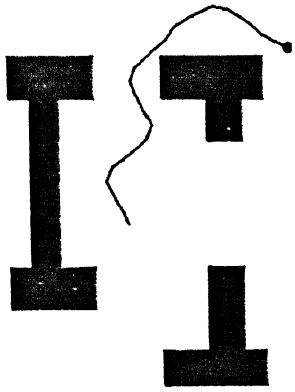


Fig.4.6(k)

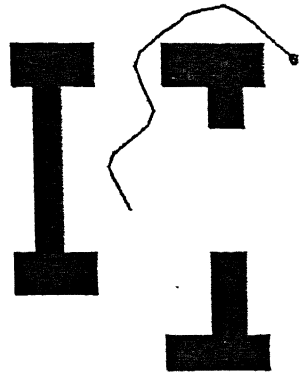


Fig.4.6(l)

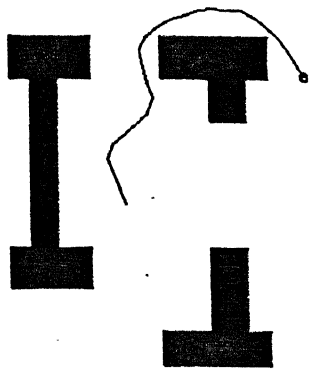


Fig.4.6(m)

Case 2 : initial position = (0,20), goal position = (-22, 13), The shortest path obtained by visibility graph search method is : $(0,20) \rightarrow (-5,25) \rightarrow (-19,25.8) \rightarrow (-22,13)$. The rest of the data is same as in case 1. This case shows the success of path planning of the manipulator using priority based method, while the same case showed failure when we applied the potential field approach (see Case 2, Fig. 3.4).

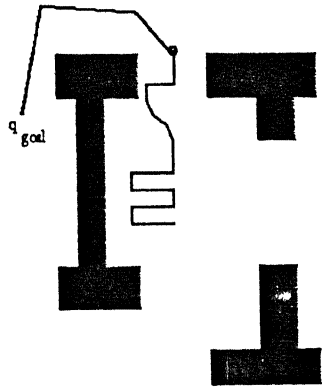


Fig.4.7(a)

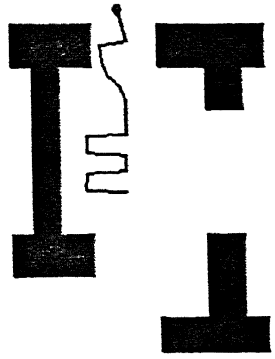


Fig.4.7(b)

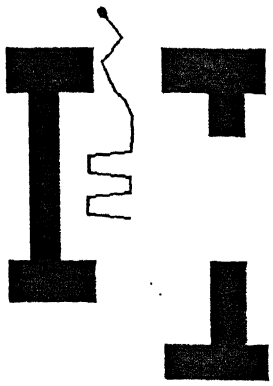


Fig.4.7(c)

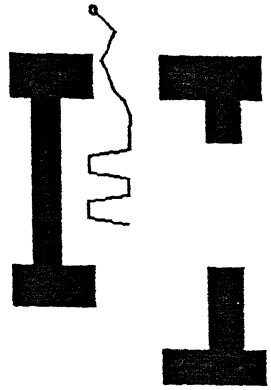


Fig.4.7(d)

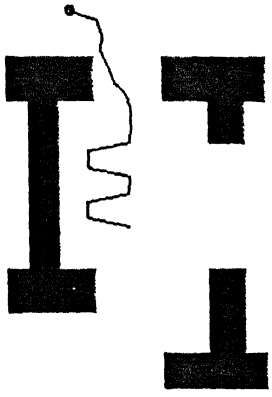


Fig.4.7(e)

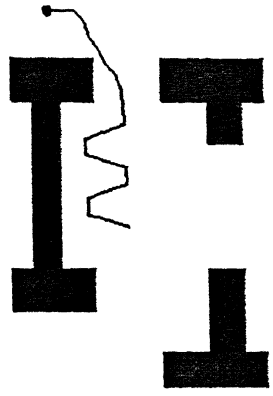


Fig.4.7(f)

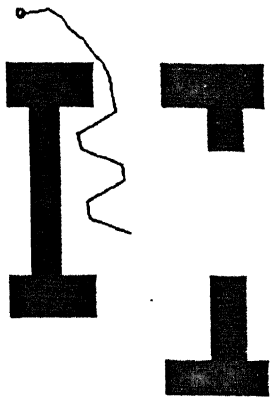


Fig.4.7(g)

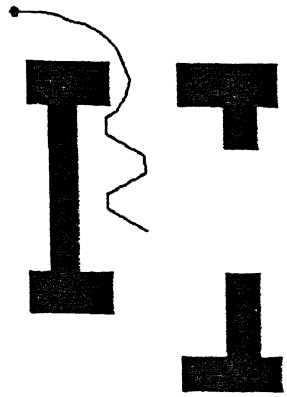


Fig.4.7(h)

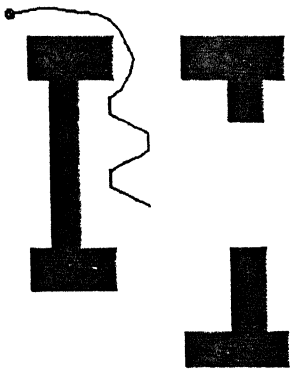


Fig.4.7(i)

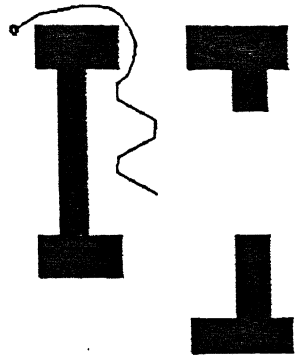


Fig.4.7(j)

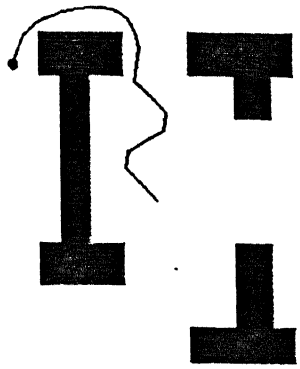


Fig.4.7(k)

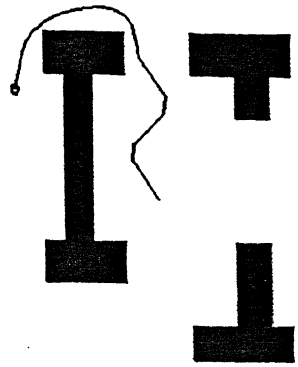


Fig.4.7(l)

Case 3 : initial position = (0,20), goal position = (20, -25), The shortest path obtained by visibility graph search method is : $(0, 20) \rightarrow (1.175, -21.475) \rightarrow (20, -25)$. The rest of the data is same as in case 1. This case shows the success of path planning of the manipulator using priority based method, while the same case showed failure when we applied the potential field approach (see Case 6, Fig. 3.8).

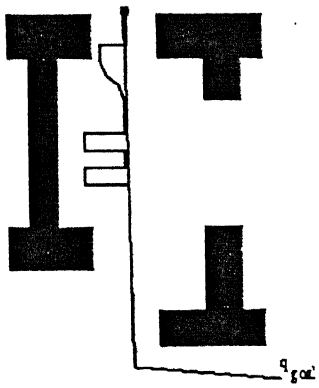


Fig.4.8(a)

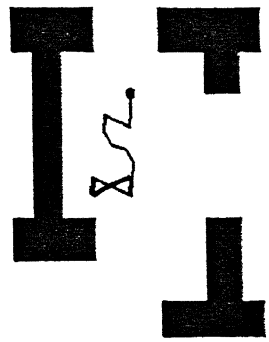


Fig.4.8(b)

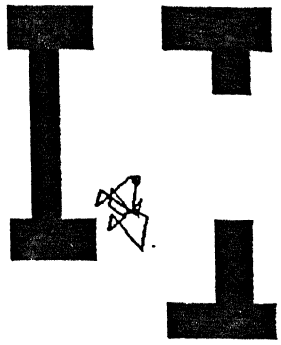


Fig.4.8(c)

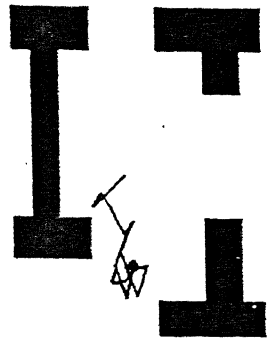


Fig.4.8(d)

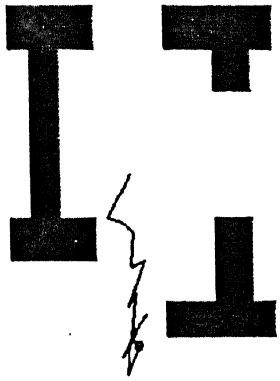


Fig.4.8(e)

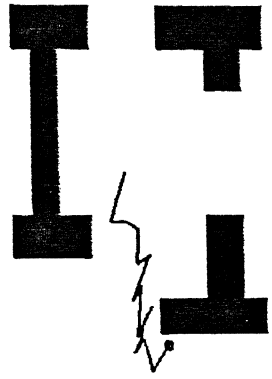


Fig.4.8(f)

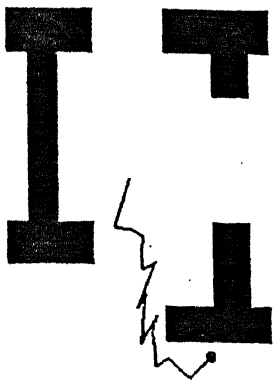


Fig.4.8(g)

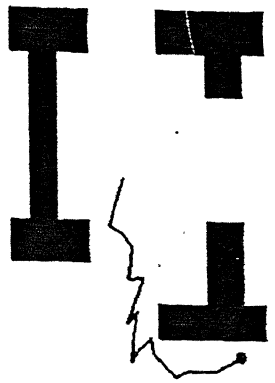


Fig.4.8(h)

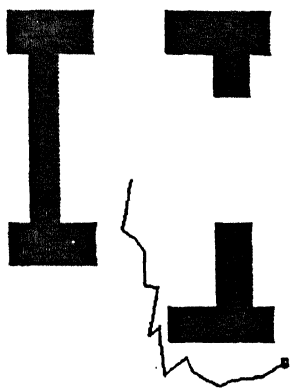


Fig.4.8(i)

Case 4 : initial position = (0,20), goal position = (-27, 7), The shortest path obtained by visibility graph search method is : $(0, 20) \rightarrow (-4, 1.4) \rightarrow (-11, 1.4) \rightarrow (-16, 23.8) \rightarrow (-23, -23.8) \rightarrow (-27, -7)$. The room dimensions are taken to be a square one having each side 35 units. The robot manipulator base is kept at (0, 0). The number of links in the hyper-redundant manipulator are taken as 36 links each having length 1 units. The initial configuration of manipulator is as shown in Fig. 2.13(a). In Fig. 2.13(a), the end-effector position is shown by a thick point which is placed at (0, 20). Here the co-ordinate

frame is fixed at the centre of room and the base of the manipulator is kept at origin.

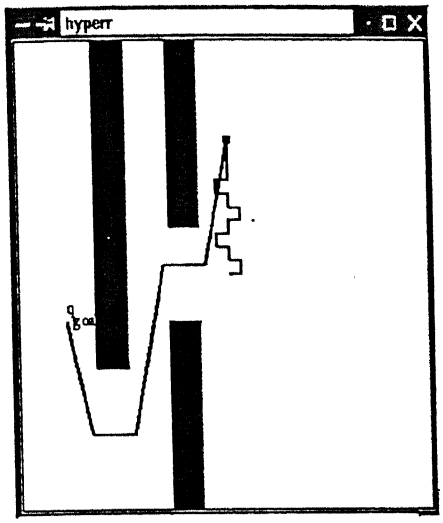


Fig.4.9(a)

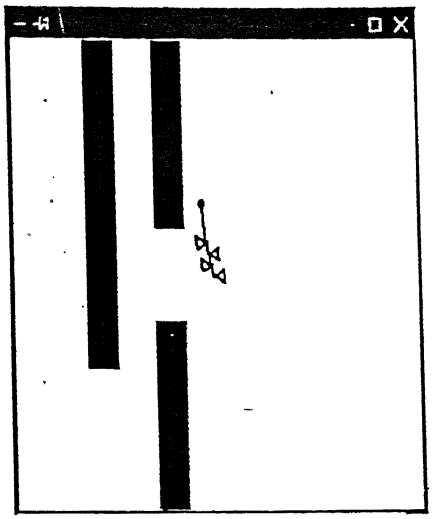


Fig.4.9(b)

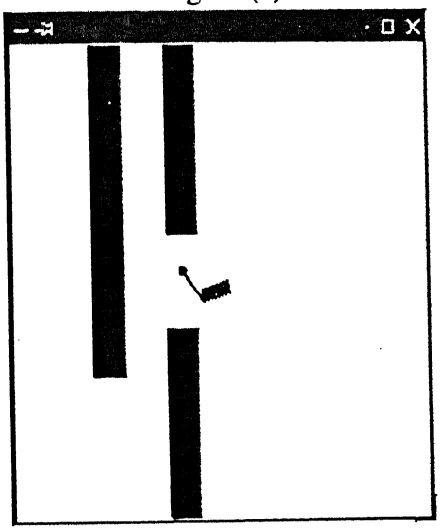


Fig.4.9(c)

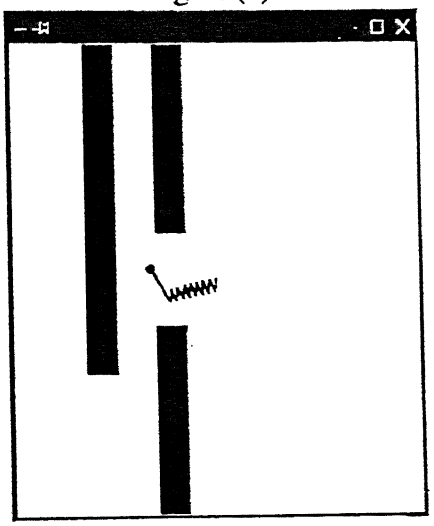


Fig.4.9(d)

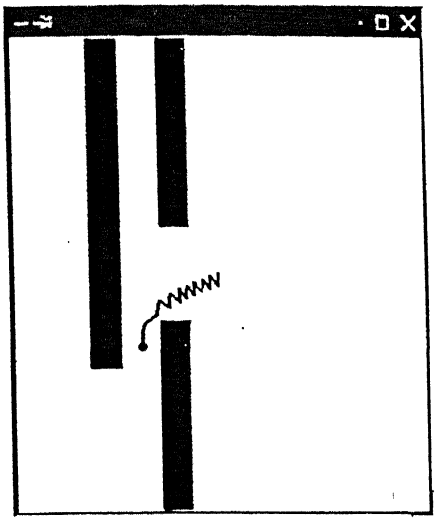


Fig.4.9(e)

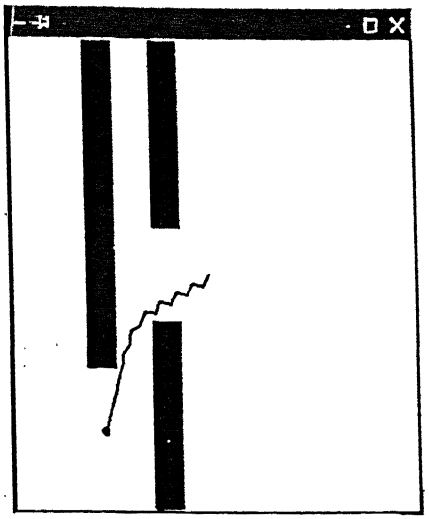


Fig.4.9(f)

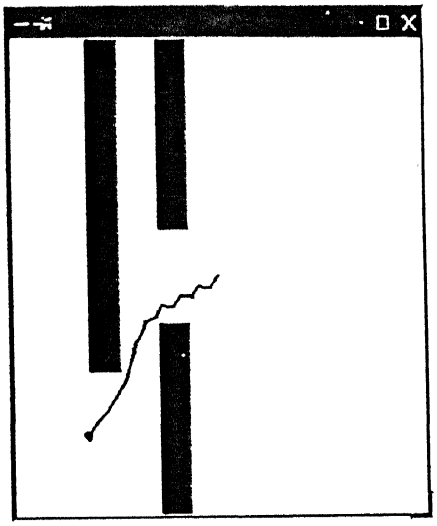


Fig.4.9(g)

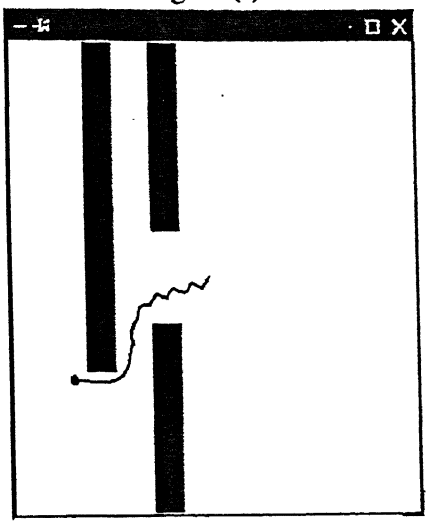


Fig.4.9(h)

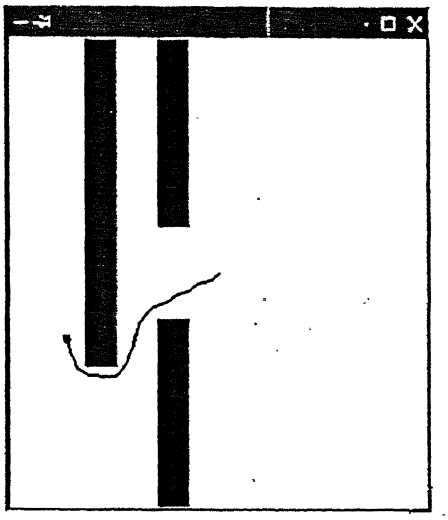


Fig.4.9(i)

4.5 Conclusion

The concept of *task priority* has been used into the inverse kinematics of redundant manipulator. We confirmed that dividing a task into subtasks with order of priority is useful not only for actively utilizing the kinematic redundancy, but also for relaxing the shortage or the degeneracy of DOF. The results show its extent of success while planning a path in cluttered environment. The cases 3 & 4 taken up under section 4.4, shows the path planning with self-intersection amongst the links of the manipulator, this can be avoided as mentioned in section 3.5.

Chapter 5

Synthesis and Optimization of Hyper-Redundant Manipulators

5.1 Manipulator Design

In chapters 3 & 4, we have implemented path planning schemes for the hyper-redundant manipulator to reach the target locations by obstacle avoidance. Now as a next step, in this chapter we have implemented a synthesis procedure which, for a given work-cell with known obstacles and known target poses, designs the manipulator. This phase involves the determination of the number of links (degrees of freedom), their relative proportion (link lengths) and kinematic arrangement.

5.2 System Overview

We follow a procedure presented in Leger [11] for synthesizing and optimizing robot kinematic parameters. This procedure uses the expertise developed in the path planning procedure. The path planning procedure, with lots of constraints due to obstacles in the work-cell sometimes does not completely result in success in reaching to the goal. The amount of failure can be accounted mainly

due to insufficient length of the manipulator to reach the goal. We use an optimizing algorithm based on genetic algorithm. We take a task-oriented approach to synthesize in which manipulator is evaluated on a designer-specified task in simulation; extensive use of path planning algorithm is thus required so that a wide variety of manipulator can be evaluated.

5.3 Genetic Algorithm

The Genetic Algorithm (GA) [12] is based on two biological phenomena: natural selection and sexual recombination. In this the possible solutions to a problem are represented as bit strings. The GA maintains a population of bit strings, and reproduces them preferentially based on quality of solutions they represent. This is anologous to natural selection. Reproduction is performed by the crossover operator, which produces an offspring solution by combining parts of two parent solutions. This simulates sexual recombination. The quality of solutions in this population improves over time, since good solutions are reproduced more frequently then bad solutions. The other general class of genetic operations is mutation. Mutation plays a small but important role in genetic methods: it prevents the population from becoming completely uniform. As the optimization process progresses, the population improves by converging on a relatively small area of the search space. This necessarily means that the population becomes less diverse. Mutation introduces small changes which allow the optimization to explore itself slightly different, and possibly better. When the population is highly fit (or well adapted to the task at hand), most mutations are likely to be detrimental. However they can expand the search space beyond what would be explored by crossover alone, and this can occasionally be beneficial. Mutation is not very useful in the early stages of optimization since the population is often very diverse.

Genetic approach does not need to rely on problem-specific features. They have two basic requirements: possible solution should be represented in a way such that two parents can be combined to yield an offspring, and a method of evaluating solutions must be supplied. These properties make genetic algorithm attractive for synthesis.

5.4 Representation

The genetic optimization process requires a suitable representation for the objects being optimized – robot kinematic parameters, in this case. We represent a manipulator configuration as a set of connected modules. Each module represents a part of the manipulator; a module might be a link, a joint, an entire arm or any other component of the manipulator. It is important to note that the modules are used only for synthesis.

Each module may have an arbitrary number of parameters which describe properties of the module. Most parameters describe the various dimensions of a particular module type, but the parameters may also be used to describe the non-kinematic properties such as the size of an actuator or the thickness of a link's structural members. This allows arbitrary properties of a module to be modified and evaluated. Each parameter has a number of components. The minimum and maximum values can be specified by the designer to limit the variation in a parameter. A *constant flag* can be set to indicate that the parameter value should not be modified by any genetic operations. Finally, each parameter has a bit-string value (used by genetic operators) and a floating point value (the actual value of the property represented by the parameter).

In this work, the manipulator links have been taken as the modules, and only link-length describes the property of a link, i.e module. No other properties such as link width, size of actuator, have been taken into consideration while synthesizing the work. Moreover, no joint has been taken as module. So the configurations are composed of a set of connected modules, i.e links. Each configuration is represented as a list of modules. Each module can specify a connection to a module that comes later in the list. In more specific terms, the configuration is stored as a topologically stored, directed acyclic graph. Each node in the graph is a module, each edge is a connection between modules.

5.5 Problem Formulation

Here, we have modelled our problem as unconstrained optimization problem. In this problem formulation we define a objective function as a function of kinematic parameters such as link-lengths and joint-angles, of the manipulator. The link-lengths are taken as the variables and the joint angles are taken

as the angles, which come from the last configuration of the manipulator in the incomplete path planning. Joint angles are treated as constants throughout the synthesis work.

The **objective function** is to **minimize** $\sum_{i=1}^{n} \mathbf{f}(l_i, \theta_i, \mathbf{r}_{goal})$, where n represents the number of variables, the variable, l_i represents the i-th link-length of a manipulator with link-length limits $LB \leq l_i \leq UB$. Where LB and UB, respectively represent lower limit and upper limit of the length of links of the manipulator. The other parameters in the objective function are constant that means joint angles $\boldsymbol{\theta}$ and the goal position $\mathbf{r}$ are treated as constants.

The objective function is a **distance** function between the last location, that has been reached by the end-effector in the incomplete planning process and the goal position to be reached. The point A in Fig. 5.1 denotes the last position of the manipulator end-effector point in the incomplete path planning. The point B shows the goal position to be reached. So our objective function to be minimized is the distance between point A and point B, so that in end of the synthesis, we should get either the corresponding changed values of the link-lengths or the number of links to be added, for which the manipulator reaches the goal.

5.6 Results

The inputs for the genetic system is the that configuration of the manipulator where it stops and fails to reach the goal location.

For **Case 1** the goal location, which is to be reached, is at point B (-22,5). The manipulator while following the path (shown by black polyline, starting from point (0,20)) stops at point A (-21.1,11) at the configuration shown in Fig. 5.1. That means with the current kinematic parameters, i.e. link-lengths and total number of links, the manipulator is unable to reach the goal. So now we feed this end configuration of the manipulator in the genetic system and ask the system to give the changed values of either link-lengths or the total number of links, so that finally with those values of the kinematic parameters the manipulator reaches the goal.

The configuration of manipulator for which it fails to reach to goal is shown in Fig 5.1. The data that is to be given as inputs to the genetic system is shown in Table. 5.1.

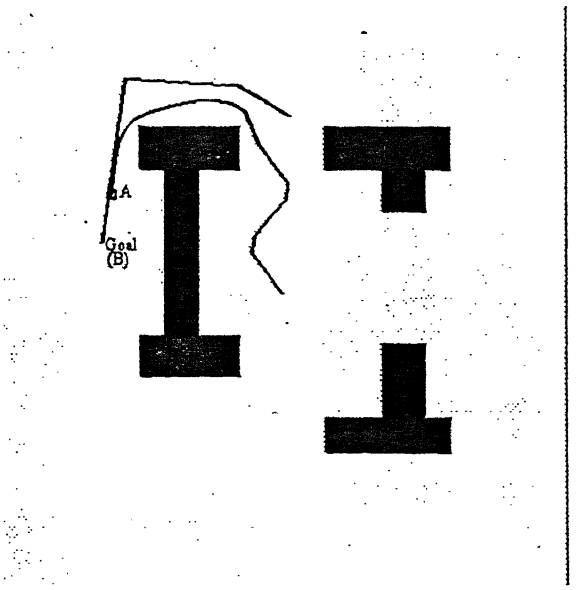


Figure 5.1: Case 1 : The configuration of the manipulator when it fails to reach the goal point at B

Link Number	1	2	3	4	5	6	7	8	9
Joint Angles (in degrees)	126.58	1.08	-0.672	-43.10	-34.66	0.105	-0.279	34.7	43.5
Link Number	10	11	12	13	14	15	16	17	18
Joint Angle (in degrees)	0.929	-2.46	-19.6	8.32	32.67	21.65	13.52	6.21	4.41
Link Number	19	20	21	22	23	24	25		
Joint Angle (in degrees)	4.06	5.58	22.9	17.63	16.21	4.72	1.22		

Table 5.1: Case 1: inputs for the genetic optimization algorithm

input	Value
No. of constraints	0
No. of variables	25
Population size	250
Chromosome length	25
No. of generations	250
Distribution index	3
Max. Perturbation	.015
Min. limit of a variable	1.6
Max. limit of a variable	2.7
Cross-over probability	.91
Mutation probability	.01
Random seed	.32

Table 5.2: Case 1: Values of genetic operators and variables

where each link has the link-length 2 units. The joint angles are in degrees and each successive joint angle is calculated with respect to its previous link-axis. These joint angles correspond to the configuration of manipulator shown in Fig. 5.1. Now this data is supplied to the genetic algorithm and the algorithm is asked to minimize the distance **AB**. The algorithm uses the data shown in Table 5.2.

Here the variables are the link-lengths of the manipulator and that is 25 in number for a 25-link manipulator. This genetic system does not give the best solution in the first iteration; instead we need to play with the values of Distribution index, Cross-over probability, Mutation probability and Random seed, till we get the best fitted solution. Here the distribution index range has been put up between .5 and 6. Similiraly cross-over probability range is between 0.5 and 1 and Mutation probability is in between 0 and 0.2.

The output from the genetic algorithm, which minimizes the distance AB is shown in Table. 5.3.

The corresponding configuration of the manipulator, after synthesis, is shown in Fig. 5.2. where the final values of manipulator link-lengths are taken, as given in the above table. Fig. 5.2 clearly shows that with these values of link lengths the manipulator reaches the goal.

Similarly, now for **Case 2** there are 3 obstacles in room as shown in Fig. 5.3. And the path from visibility graph algorithm comes out to be $(0, 20) \rightarrow (6.67, 21.34) \rightarrow$

Link Number	1	2	3	4	5	6	7	8	9
Link-length (variable)	1.937	1.905	2.07	1.96	1.87	1.87	2.09	1.65	2.03
Link Number	10	11	12	13	14	15	16	17	18
Link-length (variable)	1.94	2.02	2.06	1.75	1.85	2.19	2.10	1.97	2.36
Link Number	19	20	21	22	23	24	25		
Link-length (variable)	2.59	2.27	2.68	2.68	2.62	2.69	2.69		

Table 5.3: Case 1: Optimized values of variables after synthesis

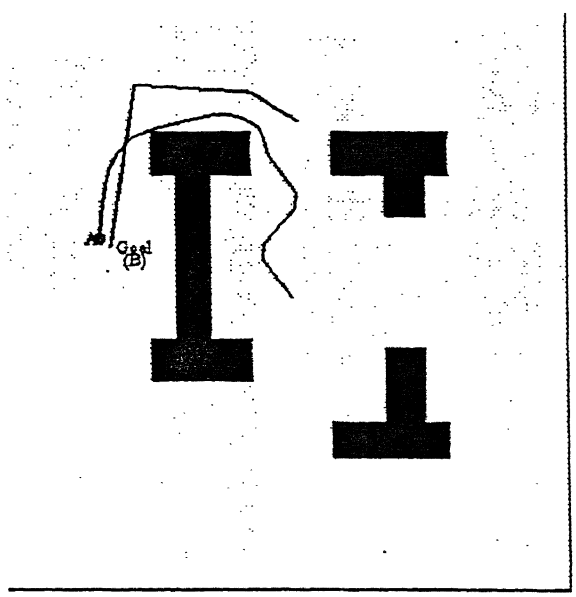


Figure 5.2: Case 1 : The configuration of the manipulator when it reaches the goal point at B, with changed values of its link-lengths after optimization

$(19.47, 21.34) \rightarrow (21, -5)$. Where (21,-5) is the goal location, i.e the coordinates of point **B**. The manipulator gets stuck at location (21.7,2.02), i.e. point **A**. Fig. 5.3 shows the configuration of the manipulator when it stops and fails to reach to the goal. So now the inputs for the GA are shown in Table. 5.4.

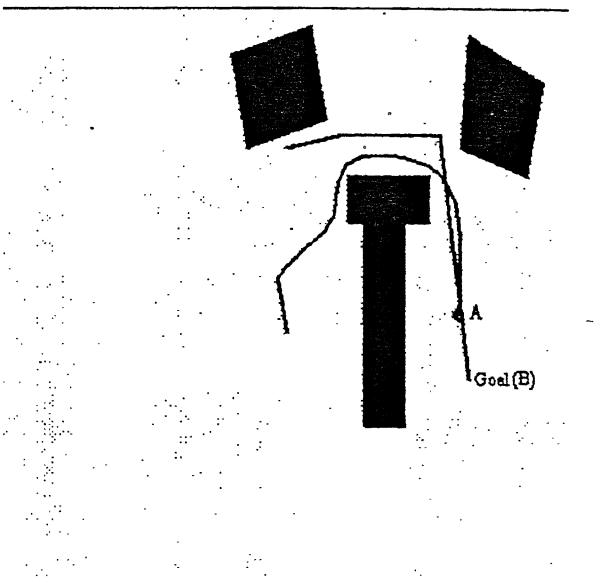


Figure 5.3: Case 2 : The configuration of the manipulator when it fails to reach the goal point at B

The inputs for the genetic system to solving this problem are given in Table. 5.5.

Here the range for Distribution index, Cross-over probability and mutation probability have been set as the same as it was for **Case 1**. The optimized values of the variables (link-lengths) after synthesis are shown in Table. 5.6.

It is shown in Fig. 5.4, as how successfully the manipulator reaches the goal location.

Link Number	1	2	3	4	5	6	7	8	9
Joint Angles (in degrees)	85.5	3.59	-4.19	-22.92	-9.03	4.303	-3.22	9.76	4.49
Link Number	10	11	12	13	14	15	16	17	18
Joint Angle (in degrees)	-6.40	-30.14	-26.34	-7.10	-2.22	-8.53	-7.131	-11.63	-30.07
Link Number	19	20	21	22	23	24	25		
Joint Angle (in degrees)	-19.72	-5.47	-1.96	-1.26	-0.76	-0.46	-0.33		

Table 5.4: Case 2: Inputs for Genetic optimization algorithm

input	Value
No. of constraints	0
No. of variables	25
Population size	250
Chromosome length	25
No. of generations	250
Distribution index	4.5
Max. perturbation	.015
Min. limit of a variable	1.6
Max. limit of a variable	2.7
Cross-over probability	.93
Mutation probability	.0151
Random seed	0.505

Table 5.5: Case 2: Input values of genetic oprators

Link Number	1	2	3	4	5	6	7	8	9
Link-length (variable)	2.31	2.20	1.87	2.27	2.02	1.98	2.03	1.79	2.22
Link Number	10	11	12	13	14	15	16	17	18
Link-length (variable)	2.08	2.03	2.15	1.77	2.53	2.54	2.43	2.188	2.07
Link Number	19	20	21	22	23	24	25		
Link-length (variable)	2.60	2.62	2.65	2.69	2.69	2.67	2.69		

Table 5.6: Case 2: Optimized values of the variables

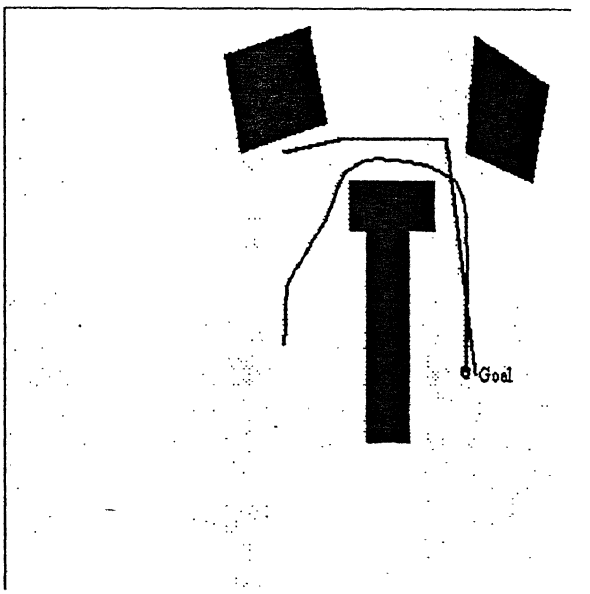


Figure 5.4: Case 2 : The configuration of the manipulator when it reaches to the goal point at B

5.7 Conclusion

Genetic approaches are very flexible in theory but in practice their flexibility is limited by the representation and evaluation used in the optimization process. We have developed a framework for synthesis of a manipulator. By combining the genetic optimization with path planning method (chapter 4). We have configured the optimization problem as an un-constrained optimization problem. This optimization technique makes extensive use of the planning algorithm. So if planner is poorly designed or is not free of bugs, it may cause an acceptable design to perform poorly. So looking at the results we can infer that the planning algorithm works reasonably good for most of the cases and if we add more modules and constraints in this optimization technique we are expected to get much better results.

5.8 Future Work

In this work, the problem has been modelled in 2-D workspace, so the very first extension is to plan and design the hyper-redundant manipulator in 3-D

workspace. In the case of 3-D work-cell, the obstacles would be modelled as polyhedrons, instead of polygons. The same planning method, i.e. *visibility graph methods*, can be used to compute the path of manipulator in 3-D workspace. then there after we can successfully apply *Order of priority* method to let the manipulator follow the path in the same fashion as it has been implemented in this work.

For the manipulator to work in 3-D workspace, we need to initiate a novel type of data structure for kinematic structures that efficiently supports intersection queries. We have formulated a model that captures a variety of settings ranging from collision detection for articulated robot arm to conformation search in planning. Here it would be worthwile mentioning that in some applications, particularly in randomized path planning (see chapter 3), there is a considerable flexibility in the order in which the operations are performed on the kinematic data structure. This raises the issue of extending the off-line results to the case where the algorithm can at least partially reorder the sequence of operations so as to minimize the total computational cost.

In task priority methods, the obstacle-avoidance problem was solved in two different ways: (1) the potential function of the obstacles was used to determine the motion of the second manipulation variable, and (2) the reference initial configuration given by an operator, based on global judgement. If we accept the human operator intervention, which can give direct informations to robot manipulator about how to avoid obstacles, then it would be much more computationally less expensive.

In an attempt to make the manipulator more dexterous in its reachability to most of the goal locations in the work-cell, we have presented genetic optimization process. We represent a robot configuration as a set of connected modules, each representing a part of robot manipulator. We have taken a link as module, and the link-lengths as their parameters to describe its property. We can extend the representation of modules by incorporating a joint, an entire arm or any other component of a manipulator as modules. We can further extend the analysis by adding the properties of a module by adding some more non-kinematic properties such as thickness of link and size of the actuator.

A dynamic simulator [6] will allow us to model the effects of actuator saturation. If a manipulator has actuators which are undersized, its accuracy and speed will be inferior to a manipulator with optimally sized actuators. On the other hand, a robot with over sized actuators will be more expensive, and each over-

sized actuator may require that other actuators be made larger. An over-sized actuator near the end of a manipulator will put greater demands on actuators near the base of the serial chain, thus either degrading performance or requiring other actuators to be enlarged. By modelling the effects of actuator size, the system is expected to optimize the size of actuators in the same way in which the kinematic dimensions are optimized. Another non-kinematic property that can be modelled to optimize is the cross-section of a manipulator links.

In our work, for simplification we have modelled the problem as un-constrained optimization problem. We can further model this problem as constrained optimization problem. In that, we can restrict the hitting of the manipulator with the obstacle in the work-cell, in the optimization process itself and consequently we will then get better possible configuration of the manipulator after synthesizing for its link-lengths.

Appendix A

A.1 Configuration

We consider a rigid object $\mathcal{A}$ — the robot — moving in a physical workspace $\mathcal{W}$ see Fig A.1. We represent $\mathcal{W}$ as the N-dimensional Euclidean space R^N. Here N = 2 equipped with a fixed cartesian system, or frame, denoted by $\mathcal{F}_\mathcal{W}$. we represent $\mathcal{A}$ at a reference position and orientation as a compact subset of R^N. A moving frame $\mathcal{F}_\mathcal{A}$ attatched to $\mathcal{A}$ so that each point in robot has fixed co-ordinates in $\mathcal{F}_\mathcal{A}$. The origins of $\mathcal{F}_\mathcal{W}$ and $\mathcal{F}_\mathcal{A}$ are denoted by $\mathcal{O}_\mathcal{W}$ and $\mathcal{O}_\mathcal{A}$ respectively. So a configuration q of $\mathcal{A}$ is a specification of the position and orientation of $\mathcal{F}_\mathcal{A}$ with respect of $\mathcal{F}_\mathcal{W}$. The *configuration space* of $\mathcal{A}$ is the space C of all the possible configurations of $\mathcal{A}$.

A configuration q of $\mathcal{A}$ can be bijectively represented, in a way that depends upon the choice of the frames $\mathcal{F}_\mathcal{A}$ and $\mathcal{F}_\mathcal{W}$, as a pair $(\mathcal{T}, \Theta)$, where $\mathcal{T}$ determines the position of the origin $\mathcal{O}_\mathcal{A}$ of $\mathcal{F}_\mathcal{A}$ in $\mathcal{F}_\mathcal{W}$ and Θ determines the orientation of $\mathcal{F}_\mathcal{A}$'s axis with respect to $\mathcal{F}_\mathcal{W}$.

Let us adopt the following convention: $\mathcal{T}$ is equal to N-vector of the co-ordinates of $\mathcal{O}_\mathcal{A}$ in $\mathcal{F}_\mathcal{W}$ and Θ is equal to $N \times N$ matrix whose columns are the components of the unit vector along $\mathcal{F}_\mathcal{A}$'s axis in $\mathcal{F}_\mathcal{W}$. Θ belongs to the so-called Special Orthogonal Group of the $N \times N$ matrices in R^{N^2} with the orthogonal columns (and rows) and determinant +1. This group is denoted by $SO(N)$.

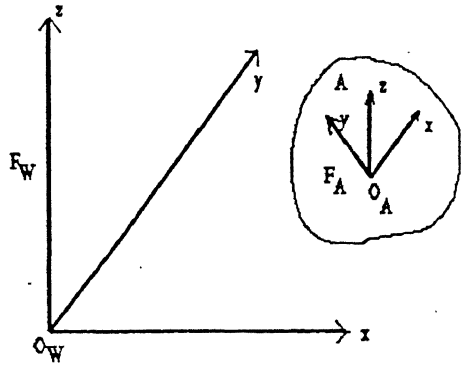


Figure A.1: The robot moves in a workspace $\mathcal{W} = \mathcal{R}^N$, with N = 2 , $\mathcal{A}$ is modelled as a compact subset of R^N. A fixed cartesian frame $\mathcal{F}_\mathcal{W}$ is embedded in $\mathcal{A}$. The configuration of $\mathcal{A}$ specifies the position and orientation of $\mathcal{F}_\mathcal{A}$ with respect to $\mathcal{F}_\mathcal{W}$.

A.2 Obstacles

Let $\mathcal{W}$ contains fixed obstacles $B_i, i = 1, 2, \ldots, q$, which we represent as closed, but not necessarily bounded [1], regions of R^N. The B_i's denote both the "physical" obstacles and the subsets of R^N that represent them.

A.3 C-Obstacle

Definition : The obstacle B_i in $\mathcal{W}$ maps in C to region $CB_i = \{q \in C / \mathcal{A}(q) \cap B_i \neq \phi\}$. CB_i is called **C-obstacle**. The union of all C-Obstacles:

$$\bigcup_{i=1}^{q} CB_i$$

is called the C-**obstacle region**, and the set:

$$C_{free} = C \setminus \bigcup_{i=1}^{q} CB_i = \{q \in C / \mathcal{A}(q) \cap (\bigcup_{i=1}^{q} B_i) \neq \phi\}$$

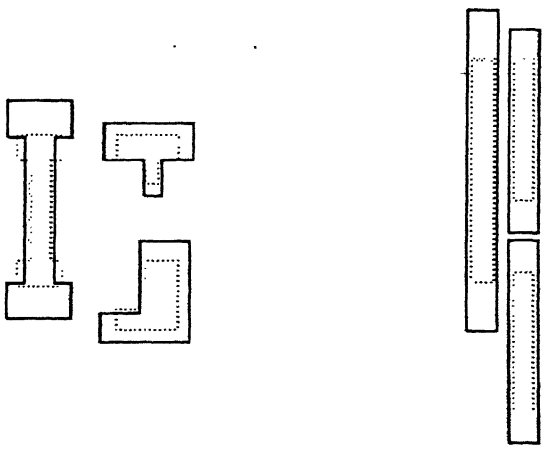
is called the **free space**. Any configuration in C_{free} is called the **free configuration**.

A **free path** between two free configurations q_{init} and q_{goal} is a continous map $\tau : [0,1] \rightarrow C_{free}$, with $\tau(0) = q_{init}$ and $\tau(1) = q_{goal}$. Two configurations belong to the same connected component of C_{free} if and only if they are connected by a free path.

Given an initial and a goal configuration, the basic motion planning problem is to generate a free path between the two configurations, if they belong to the same connected component of C_{free}, and to report failure otherwise.

A.4 Semi-Free Path

A **semi-free path** is a continuous map $\tau : [0,1] \rightarrow cl(C_{free})$, where $cl(C_{free})$ denotes the closure of C_{free}. Hence, as it moves along such a path, the robot may touch obstacles. Under simplifying assumptions any semi-free path τ can be transformed into a free path that can be made arbitrarily close to τ.



(a) (b)

Figure B.2: Two examples : (a) The dark colour representation shows the final enlarged form of the obstacles shown in light colour. (b) Another result.

Appendix C

C.1 Pseduoinverse

Eq. 2.2 in chapter 2 represents the instantenous kinematic equation for a general n degrees of freedom manipulator arm. Where the dimension of the manipulator Jacobian $\boldsymbol{J}$ is $m \times n$. When n is larger than m and $\boldsymbol{J}$ is of full rank, there are $(n - m)$ arbitrary variables in the general solution of Eq. 2.2. The manipulator is then said to have $(n - m)$ *redundant degrees of freedom* for a given task.

The Jacobian matrix also determines the relationship between the end-effector velocity $\dot{\boldsymbol{p}}$ and joint velocities $\dot{\boldsymbol{q}}$:

$$\dot{\boldsymbol{p}} = \boldsymbol{J}\dot{\boldsymbol{q}} \tag{C.1}$$

In this section we would discuss n optimal solution to the velocity relationship (C.1). We fix the manipulator Jacobian at an appropiate arm configuration, and find the optimal solution to the linear equation (C.1), assuming that the jacobian matrix is of full row-rank. We evaluate the solutions to the linear equation by the quadratic cost function of joint velocity vector given by :

$$G(\dot{\boldsymbol{q}}) = \dot{\boldsymbol{q}}^T \boldsymbol{W} \dot{\boldsymbol{q}} \tag{C.2}$$

where W is an $n \times n$ symmetric positive definite weighting matrix. The problem is to find the $\dot{\boldsymbol{q}}$ that satisfies the equation (C.1) for a given $\dot{\boldsymbol{q}}$ and $\boldsymbol{J}$ while

minimizing the cost function $G(\dot{q})$. Let us solve this problem using Lagrange multipliers. To this end we use a modified cost function of the form :

$$G(\dot{q}, \lambda) = \dot{q}^T W \dot{q} - \lambda^T (J\dot{q} - \dot{p}) \tag{C.3}$$

where λ is am $m \times 1$ unknown vector of Lagrange multipliers. The necessary conditions that the optimal solution must satisfy are

$$\frac{\partial G}{\partial \dot{q}} = 0, \quad i.e. \quad 2W\dot{q} - J^T\lambda = 0 \tag{C.4}$$

and

$$\frac{\partial G}{\partial \lambda} = 0, \quad i.e. \quad J\dot{q} - \dot{p} = 0 \tag{C.5}$$

which is ofcourse identical to equation (C.1). Now the matrix W is positive definite, hence invertible. Thus we obtain from equation (C.4)

$$\dot{q} = \frac{1}{2} W^{-1} J^T \lambda \tag{C.6}$$

Substituting the above into (C.5) yields

$$(JW^{-1}J^T)\lambda = 2\dot{p} \tag{C.7}$$

Since J is assumed to be of full rank, matrix product $JW^{-1}J^T$ is a full-rank square matrix, and is therefore invertible. Eliminating the Lagrange multiplier vector λ in equation (C.6) and (C.7), we obtain the optimal solution as

$$\dot{q} = W^{-1}J^T(JW^{-1}J^T)^{-1}\dot{p} \tag{C.8}$$

Clearly, the above solution satisfies the original velocity relationship (C.1). Indeed we can obtain equation (C.8) by Jacobian matrix J. When the weighting matrix W is the $m \times m$ identity matrix, the above solution reduces to

$$\dot{q} = J^T(JJ^T)^{-1}\dot{p} \tag{C.9}$$

The matrix product $J^{\#} = J^T(JJ^T)^{-1}$ is known as *pseudo-inverse* of the Jacobian matrix.

Bibliography

[1] J. C. Latombe – *Robot Motion Planning.* Kluwer Academic Publishers, USA.

[2] Johan J. Craig – *Introduction To Robotics.* Addison-Wesley Publishing Company, Inc.

[3] H. Asada And J.-J. E. Slotine – *Robot Analysis And Control.* John Wiley And Sons, pp. 66-71.

[4] Thomas H. Cormen, Charles E. Leiserson, And Ronald L. Rivest – *Introduction To Algorithm.* Printics Hall Of India, pp. 465-488.

[5] Y. Nakamura – *Advanced Robotics Redundancy And Optimization.* Addison-Wesley Publishing Company, Inc., pp. 2-11, pp. 125-150.

[6] D. Halperin, J. C. Latombe, And R. Motwani – *Dynamics Maintenance of Kinematic Structures.* Full version (1996), *http://theory.standford.edu/people/rajeev/postscrits/dynamic.ps.Z.*

[7] J. R. Singh And J. Rastegar – *Optimal Synthesis of Robot Manipulators Based on Global Kinematic Parameters. ASME J.* Vol. 30. No. 4, pp. 569-580, 1995.

[8] Y. Koga And J. C. Latombe – *On Multi-Arm Manipulation Planning. Proc. IEEE Int. Conf. Robotics And Automation,* Nice, France 1995.

[9] K. Goldberg, D. Halperin, J. C. Latombe, And R. H. Wilson (eds). – *Algoritmic Foundations of Robotics.* AkPeters Welleseley MA, 1995.

[10] H. Choset, W. Henning *A Follow-The-Leader Approach to Serpentine Robot Moion Planning Proc. IEEE Int. Conf. on Robotics and Automation*

[11] C. Leger – *Automated Synthesis And Optimization of Robot Configurations*. Ph. D. Thesis Proposal. Carnegie Mellon University, 1997. Available on the WWW from *http://www.frc.ri.cmu.edu/ blah/papers/*.

[12] K. Deb – *Optimization For Engineering Design*. Prentice Hall India, pp. 290-320.